# GE Fanuc Automation

***CIMPLICITY® Monitoring and Control Products***

# *CIMPLICITY HMI Plant Edition*

# *Tracker*

## *Production Tracking Operation Manual*

*GFK-1216F* *July 2001*

**Following is a list of documentation icons:**

**Warning** notices are used to emphasize that hazardous voltages, currents, temperatures, or other conditions that could cause personal injury exist in the equipment or may be associated with its use.

**Caution** provides information when careful attention must be taken in order to avoid damaging results.

**Important** flags important information.

**To do** calls attention to a procedure.

**Note** calls attention to information that is especially significant to understanding and operating the equipment or software.

**Tip** provides a suggestion.

**Guide** provides additional directions for selected topics.

This document is based on information available at the time of publication. While efforts have been made to be accurate, the information contained herein does not purport to cover all details or variations in hardware or software, not to provide for every possible contingency in connection with installation, operation, or maintenance. Features may be described herein which are not present in all hardware and software systems. GE Fanuc Automation assumes no obligation of notice to holders of this document with respect to changes subsequently made.

GE Fanuc Automation makes no representation of warranty, expressed, implied, or statutory with respect to, and assumes no responsibility for the accuracy, completeness, sufficiency, or usefulness of the information contained herein. No warranties of merchantability or fitness for purpose shall apply.

CIMPLICITY is a registered trademark of GE Fanuc Automation North America, Inc.
Windows, Windows NT, Windows 98 and Windows 2000 are registered trademarks of Microsoft Corporation

This manual was produced using *Doc-To-Help*®, by WexTech Systems, Inc.

**Copyright 1997-2001 GE Fanuc Automation North America, Inc.**
**All rights reserved**

# Preface

## Contents of this Manual

**Chapter 1. Introducing Production Tracking:** Introduces CIMPLICITY HMI Production Tracking.

**Chapter 2. Production Tracking Overview:** Provides an overview of Production Tracking capabilities and interface requirements.

**Chapter 3. Using the Production Tracking Window:** Discusses the options available in the Production Tracking window (User Interface).

**Chapter 4. Configuration File Guide:** Documents the configuration files needed for Production Tracking

**Chapter 5. Production Tracking Application Interface:** Provides information on the application subroutine interface for Production Tracking.

**Chapter 6. PRT API Macros:** Documents the macros available for Production Tracking.

**Appendix A. PRT API Error Codes:** Lists the error codes returned by Production Tracking.

**Appendix B. Event Codes for Interested Process Updates:** Documents the event codes available for interested process updates.

**Appendix C. PRT Rules and Assumptions:** Documents the general rules, and combined and dispersed items tracking region rules.

**Appendix D. Using the Global Verifier:** Instruction for using the global verifier to validate and repair global sections used by Production Tracking.

**Appendix E. Basic Control Engine Extensions for Production Tracking:** Discusses the Basic language extensions available for Production Tracking in the Basic Control Engine.

**Appendix F. Configurable Tracking Interface:** Details the information necessary to configure and use the Configurable Tracking Interface (CTI).

**Appendix G. PRT Logging Configuration:** Documents the configuration for ODBC data logging of PRT information.

**Appendix H. Configuring PRT Client Startup:** Documents the configuration files that must be modified in order to have the PRT Client application start when CIMPLICITY starts.

**Appendix I. Attribute Counts by Region:** Provides a sample Attribute Counts configuration file.

**Appendix J. Troubleshooting:** Describes the PRT Interested Process Utility.

# Related Publications

For more information, refer to these publications:

***CIMPLICITY HMI Plant Edition Tracker Getting Started Guide*** (GFK 1694)

***CIMPLICITY HMI Plant Edition Tracker Production Decision Control Operation Manual*** (GFK-1218)

***CIMPLICITY HMI Plant Edition Tracker Routing Control Objects Operation Manual*** (GFK-1480)

***CIMPLICITY HMI Plant Edition Base System User's Manual*** (GFK-1180)

***CIMPLICITY HMI Plant Edition Basic Control Engine Language Reference Manual*** (GFK-1283)

***CIMPLICITY HMI Basic Control Engine Program Editor Operation Manual*** (GFK-1305)

***CIMPLICITY HMI Basic Control Engine Event Editor and BCEUI Operation Manual*** (GFK-1282)

# Contents


## Introducing Production Tracking 1-1

Welcome to Production Tracking ... 1-1
Production Tracking Concepts ... 1-3
Tracker Server and Display Options ... 1-5
Tracker Installation ... 1-6
Tracker Option Registration ... 1-8

## Production Tracking Overview 2-1

About Production Tracking ... 2-1
- Using the Tracking Model Summary ... 2-2
- Tracking Regions ... 2-3
- Associating Items ... 2-4
- Modifying an Item's Status ... 2-5
- Monitoring Sequences ... 2-6
- Generating CIMPLICITY Alarms ... 2-8
- Configuring CIMPLICITY Setpoints ... 2-8
- Using the PRT User Interface ... 2-8
- Tracking Items ... 2-9
- Using Item Classes ... 2-10
- Generating Reports ... 2-11
- Using the PRT Interface to External Software ... 2-11
- Tracking Data ... 2-11

CIMPLICITY Software Component Interface Requirements ... 2-12

## Using the Production Tracking Window 3-1

About the Production Tracking Window ... 3-1
Procedure to Open the PRT Window ... 3-2
Procedure to Set Information Parameters ... 3-4
Common Navigation Techniques ... 3-6
PRT UI Functions ... 3-9
- Connecting to Multiple Projects ... 3-10
- Moving Items between Projects ... 3-12
- Adding, Inserting or Modifying an Item ... 3-13
- Deleting an Item ... 3-22
- Fetching an Item ... 3-23
- Moving an Item ... 3-25
- Advancing an Item ... 3-27
- Finding an Item ... 3-28
- Reordering Items ... 3-29
- Updating a Region's Status ... 3-30
- Auto Locking Regions ... 3-31
- Printing Reports ... 3-32

## Configuration File Guide 4-1


About the Configuration Files ..... 4-1
Identify the Configuration Files ..... 4-2
Use the IDTPOP Utility ..... 4-4
About Configuration File Setup ..... 4-5
Configuration File Definitions ..... 4-7
PRT_SYSDEF ..... 4-8
PRT_SERVICE ..... 4-9
PRT_INTPROC ..... 4-11
PRT_GROUP ..... 4-12
PRT_REGION ..... 4-13
PRT_GRP_REG ..... 4-17
PRT_ITEM_TYP ..... 4-18
PRT_ROUTE ..... 4-19
PRT_TYPE_TRN ..... 4-21
PRT_INV_TYPE ..... 4-22
PRT_ASSOC ..... 4-23
PRT_STATUS ..... 4-24
PRT_DISP_PT ..... 4-26
PRT_ATT_PT ..... 4-27
PRT_REG_AT ..... 4-30
PRT_ATTRIBUTES.CFG ..... 4-32
PRT_STND.CFG ..... 4-34
PRT_WINDOW.CFG ..... 4-36
PRT Client – Routing Items across Project Boundaries ..... 4-39
Automated Region Validation ..... 4-41
Sequence of Events for Region Validation ..... 4-42
Reviewing Different Scenarios for Flow of Control ..... 4-44
Accessing Configuration files by the Subsystem ..... 4-46
prt_svc_val.cfg ..... 4-47
Runtime Data ..... 4-48
PRT Region Status Bits ..... 4-50
Configuration Validation Program ..... 4-52
Production Tracking Import Program (PRT_IMPORT) ..... 4-53
Program Elements (Production Tracking Import Program) ..... 4-53
Import File Naming Conventions ..... 4-53
Import File Format ..... 4-54
Running the Import Program ..... 4-64
Production Tracking Export Program (PRT_EXPORT) ..... 4-67
Export Input File Format ..... 4-67
Export Output File Format ..... 4-69
Running the Export Program ..... 4-69
Production Tracking Processes ..... 4-70
Production Tracking Global Section Usage ..... 4-70
PRT Customization ..... 4-71
Synchronization Capability ..... 4-71
Alarm Move Exceptions ..... 4-71

## Production Tracking Application Interface 5-1


About the Production Tracking Application Interface ..... 5-1
Contents of the Application Subroutine Interface ..... 5-1
Include Files ..... 5-2
Run-Time Notes ..... 5-2
Application Subroutines ..... 5-3
prt_api_init - Initialize The PRT_API Interface ..... 5-3
prt_api_term - Terminate PRT_API Interface ..... 5-5
prt_api_get_req - Get Request From External Process ..... 5-6
prt_api_load_item - Load Serialized Item Tracking Data ..... 5-8
prt_api_load_item_hold - Load Serialized Item Hold Data ..... 5-10
prt_api_send_resp - Send Response To Production Tracking ..... 5-12
prt_api_modify_item - Modify Item Tracking Data ..... 5-13
prt_api_modify_item_att - Modify Item Tracking Data Attributes ..... 5-16
prt_api_modify_item_all - Modify Item Tracking Data (All) ..... 5-18
prt_api_group_hold - Activate/Deactivate Hold Specification for a Tracking Group ..... 5-22
prt_api_delete_item - Delete/Scrap Item From Tracking Region Queue ..... 5-23
prt_api_delete_items_at_loc - Delete/Scrap Items From Tracking Region Queue Location ..... 5-26
prt_api_locate_item - Locate Serialized Item ..... 5-28
prt_api_req_groupitemlist - Request List Of Serialized Item IDs in a Tracking Group ..... 5-31
prt_api_unload_groupitemlist - Unload List Of Serialized Item IDs in a Tracking Group ..... 5-32
prt_api_req_class_itemlist - Request List Of Item Tracking Data ..... 5-33
prt_api_req_itemlist - Request List Of Item Tracking Data ..... 5-35
prt_api_unload_itemlist - Unload List Of Items ..... 5-37
prt_api_req_regionlist - Request List Of Tracking Region Data ..... 5-39
prt_api_unload_regionlist - Unload List Of Tracking Region Data ..... 5-41
prt_api_req_grouplist - Request List Of Tracking Group Data ..... 5-43
prt_api_unload_grouplist - Unload List Of Tracking Groups ..... 5-45
prt_api_register_intproc - Register As An Interested Process ..... 5-46
prt_api_dyn_register_intproc - Register As A Dynamically Interested Process ..... 5-47
prt_api_intproc_region - Receive updates only from specific regions ..... 5-49
prt_api_intproc_retry_connect - Send INIT Segment to all data servers ..... 5-50
prt_api_get_intproc_updates - Get Interested Process Item/Region Tracking Data Updates ..... 5-51
prt_api_term_intproc - Terminate PRT_API Interested Process Interface ..... 5-54
prt_api_static_region_data - Copy Static Region Information Into Region Structure ..... 5-55
prt_api_prod_start - Initiate Production Start For An Item ..... 5-56
prt_api_prod_stop - Enable Production Stop For An Item ..... 5-59
prt_api_add_item - Add Item To Tracking Region Queue ..... 5-62
prt_api_insert_item - Insert Item Into Tracking Region Queue ..... 5-66
prt_api_modify_region - Modify Tracking Region Data ..... 5-70
prt_api_move_item - Move Item To Location In Tracking Region ..... 5-71
prt_api_move_item_to_detain - Move Item To Detainment Region ..... 5-74
prt_api_check_route - Validate Tracking Route ..... 5-76
prt_api_advance_model - Advance Tracking Model ..... 5-77
prt_api_region_setpoints - Enable/Disable Tracking Region Setpoints ..... 5-78
prt_api_reorder_region - Reorder locations within Tracking Region ..... 5-79
prt_api_next_svc - Return each successive configured PRT Service ..... 5-81

prt_api_get_item_types - Return a list of all configured item types ..... 5-82
prt_api_get_item_tracking_type - Return the tracking type for a specific item ..... 5-83
prti_get_RCMconnection - Connect to Remote Connection Manager ..... 5-84
prti_term_RCMconnection - Disconnect from Remote Connection Manager ..... 5-85
Process Resynchronization ..... 5-86

**PRT API Macros 6-1**

About PRT Macros ..... 6-1

**Appendix A - PRT API Error Codes A-1**

Error codes Returned by Production Tracking ..... A-1

**Appendix B - Event Codes for Interested Process Updates B-1**

About Event Codes ..... B-1
Item-Related Event Codes ..... B-1
Region-related Interested Process Updates ..... B-4

**Appendix C - PRT Rules and Assumptions C-1**

General Rules ..... C-1
Combine Items Tracking Region Rules ..... C-3
Disperse Items Tracking Region Rules ..... C-4
Shifting Region Tracking ..... C-6
Shifting Region Tracking Configuration ..... C-6
Shifting Region Operations ..... C-6
Adding an item into a Shifting Region ..... C-6
Inserting an Item into a Shifting Region ..... C-7
Moving an Item into a Shifting Region ..... C-8
Deleting or Moving an Item from a Shifting Region ..... C-9
Modifying an Item from a Shifting Region ..... C-10
Routing ..... C-10
Shifting Region Process Behavior ..... C-11
PRT ..... C-11
GLB_VERIFY ..... C-11
PRT_UI ..... C-11
PRT_EXPORT/ PRT_IMPORT ..... C-11
PRT_GRD ..... C-11

**Appendix D - Using the Global Verifier D-1**

About the Global Verifier ..... D-1
Validations and Repairs ..... D-2

## Appendix E - Basic Control Engine Extensions for Production Tracking E-1


PRT Object Model ..... E-1
PRT Object Attributes ..... E-2
Prt ..... E-3
PrtService ..... E-3
PrtGroup ..... E-3
PrtRegion ..... E-3
PrtItem ..... E-3
PrtAttribute ..... E-3
Prt.GetGroupList (Method ..... E-4
Prt.GetItemList (Method) ..... E-4
Prt.GetRegionList (Method) ..... E-5
Prt.GetServiceList (Method) ..... E-5
Prt.Group (Function) ..... E-6
Prt.GroupCount (Property Read) ..... E-6
Prt.Item (Function) ..... E-7
Prt.ItemClassId (Property Read/Write) ..... E-7
Prt.ItemCount (Property Read) ..... E-8
Prt.LocateItem (Method) ..... E-8
Prt.ProjectId (Property Read/Write) ..... E-9
Prt.Region (Function) ..... E-9
Prt.RegionCount (Property Read) ..... E-10
Prt.Service (Function) ..... E-10
Prt.ServiceCount (Property Read) ..... E-11
PrtAttribute.Id (Property Read/Write) ..... E-11
PrtAttribute.Value (Property Read) ..... E-12
PrtCheckRoute (Method) ..... E-13
PrtGroup.ClearExtHold (Method ..... E-14
PrtGroup.ExtHold (Property Read) ..... E-15
PrtGroup.ExtHoldReason (Property Read) ..... E-15
PrtGroup.GetItemList (Method) ..... E-16
PrtGroup.GetRegionList (Method) ..... E-16
PrtGroup.Id (Property Read/Write) ..... E-17
PrtGroup.Item (Function) ..... E-17
PrtGroup.ItemClassId (Property Read/Write) ..... E-18
PrtGroup.ItemCount (Property Read) ..... E-18
PrtGroup.LocateItem (Method) ..... E-19
PrtGroup.Modify (Method) ..... E-19
PrtGroup.ProjectId (Property Read/Write) ..... E-20
PrtGroup.Region (Function) ..... E-20
PrtGroup.RegionCount (Property Read) ..... E-21
PrtGroup.SetExtHold (Method) ..... E-21
PrtItem.Add (Method) ..... E-22
PrtItem.Attr (Function) ..... E-23
PrtItem.AttrCount (Property Read) ..... E-24
PrtItem.ClearModifyTime (Method) ..... E-24
PrtItem.ClearStatusBit (Property Read/Write ..... E-25
PrtItem.Comment (Property Read/Write) ..... E-26
PrtItem.Delete (Property Read/Write) ..... E-27
PrtItem.EntryTime (Property Read) ..... E-27
PrtItem.ExtHold (Property Read/Write) ..... E-28
PrtItem.ExtHoldGroupId(Property Read/Write) ..... E-29

PrtItem.ItemClassId (Property Read/Write) ..... E-29
PrtItem.ItemId (Property Read/Write) ..... E-30
PrtItem.Insert (Method) ..... E-31
PrtItem.IntHold (Property Read/Write) ..... E-32
PrtItem.IntHoldReason (Property Read/Write) ..... E-33
PrtItem.ItemTypeId (Property Read/Write) ..... E-34
PrtItem.Modify (Method) ..... E-34
PrtItem.ModifyTime (Property Read) ..... E-35
PrtItem.MoveTo (Method) ..... E-36
PrtItem.ParentId (Property Read/Write) ..... E-37
PrtItem.ProdStart (Method) ..... E-38
PrtItem.ProdStop (Method) ..... E-39
PrtItem.ProjectId (Property Read/Write) ..... E-40
PrtItem.RefId (Property Read/Write) ..... E-40
PrtItem.RegionId (Property Read/Write) ..... E-41
PrtItem.RegionLoc (Property Read/Write) ..... E-42
PrtItem.RemoveAttr(Method) ..... E-43
PrtItem.ServiceId (Property Read/Write) ..... E-43
PrtItem.SetAttr (Method) ..... E-44
PrtItem.SetStatusBit (Method) ..... E-44
PrtItem.Status (Property Read/Write) ..... E-45
PrtItem.StatusBit (Function) ..... E-45
PrtItem.UserId (Property Read/Write) ..... E-46
PrtRegion.AdvanceModel (Method) ..... E-47
PrtRegion.Capacity (Property Read) ..... E-47
PrtRegion.ClearStatusBit (Method) ..... E-48
PrtRegion.DeleteItem (Method) ..... E-49
PrtRegion.GetData (Method) ..... E-50
PrtRegion.GetItemList (Method) ..... E-51
PrtRegion.Id (Property Read/Write) ..... E-52
PrtRegion.Item (Function) ..... E-52
PrtRegion.ItemClassId (Property Read/Write) ..... E-53
PrtRegion.ItemCount (Property Read) ..... E-53
PrtRegion.ItemType (Function) ..... E-54
PrtRegion.ItemTypeCount (Property Read) ..... E-54
PrtRegion.LocateItem (Method) ..... E-55
PrtRegion.Loc (Property Read/Write) ..... E-56
PrtRegion.MaxLocation (Property Read) ..... E-57
PrtRegion.Modify (Method) ..... E-57
PrtRegion.ProjectId (Property Read/Write) ..... E-58
PrtRegion.Reorder (Method) ..... E-59
PrtRegion.SetPoints (Method) ..... E-60
PrtRegion.SetStatusBit (Method) ..... E-61
PrtRegion.Status (Property Read/Write) ..... E-62
PrtRegion.StatusBit (Method) ..... E-63
PrtRegion.StatusPointId (Property Read) ..... E-64
PrtRegion.TotalItems (Property Read) ..... E-64
PrtRegion.TotalItemsPointId (Property Read/Write) ..... E-65
PrtService.GetGroupList (Method) ..... E-65
PrtService.GetItemList (Method) ..... E-66
PrtService.GetRegionList (Method) ..... E-66
PrtService.Group (Function) ..... E-67
PrtService.GroupCount (Property Read) ..... E-67
PrtService.Id (Property Read/Write) ..... E-68
PrtService.Item (Function) ..... E-68

PrtService.ItemClassId (Property Read/Write) ..... E-69
PrtService.ItemCount (Property Read) ..... E-69
PrtService.LocateItem (Method) ..... E-70
PrtService.Region (Function) ..... E-70
PrtService.RegionCount (Property Read) ..... E-71

**Appendix F - Configurable Tracking Interface F-1**

About the Configurable Tracking Interface ..... F-1
CIMPLICITY Environment ..... F-1
CIMPLICITY Base System Configuration ..... F-2
Site Configuration ..... F-2
Configurable Tracking Interface-Specific Configuration Files ..... F-5
CTI / PRT Mapping File (CTI_RP.CFG) ..... F-6
Field Layout File (CTI_RP_LAYOUT.CFG) ..... F-6
Field Translation File (CTI_XLATE.CFG) ..... F-7
Point Definition File (CTI_PT_DEF.CFG) ..... F-8
Attribute Template File (PRT_SRV_ATTR.CFG) ..... F-11
CTI Disable Logical ..... F-12
Layout of Data Points ..... F-13
Application Startup ..... F-15
System Startup ..... F-15
CIMPLICITY for NT Startup ..... F-15
Point Updates ..... F-16
Transitions Between Regions ..... F-16
Stale Records ..... F-16
Run-Time Diagnostics ..... F-17
Debugging Logicals ..... F-17
COR_STATUS Messages ..... F-18

**Appendix G - PRT Logging Configuration G-1**

About PRT Data Logging ..... G-1
PRT Logging Format ..... G-1
Tables ..... G-1
Table Fields ..... G-2
PRT_GRP ..... G-2
PRT_ITEM ..... G-2
PRT_REG ..... G-4
Configuration Files ..... G-5
Datalog.idt ..... G-5
Data_field.idt ..... G-6

**Appendix H – Configuring PRT Client Startup H-1**

About PRT Client Startup Configuration ..... H-1
PRT Client Configuration Files to be Modified ..... H-1
Physproc ..... H-2
Logproc ..... H-2
Node_Logproc ..... H-2
Service ..... H-3
Master.mcp ..... H-3

Transition Type Overview ..... H-4
Transition Type 1. Limit Switch ..... H-4
Transition Type 2. Type Detector ..... H-5
Transition Type 3. Item Identifier ..... H-5
Transition Type 4. Reference Identifier ..... H-6
Transition Type 5. Associate Reference ID to Item ..... H-7
Transition Type 6. Associate Item Type to Item ..... H-8
Transition Type 7. Positive Edge Trigger ..... H-8
Transition Type 8. Negative Edge Trigger ..... H-9

**Appendix I – Attribute Counts by Region I-1**

About Attribute Counts by Region ..... I-1
Configuration Files ..... I-3
Setting Up ..... I-5
Logproc ..... I-5
Node_Logproc ..... I-5
Physproc ..... I-6
Prt_Intproc ..... I-6
Service ..... I-6
master.mcp ..... I-6

**Appendix J – Troubleshooting J-1**

Troubleshooting Overview ..... J-1
PRT Interested Process Utility ..... J-2
Configuring the PRT Interested Process Utility ..... J-3
Running the PRT Interested Process Utility ..... J-9
Related Publications ..... v

**Index i**

# Introducing Production Tracking

## Welcome to Production Tracking

Production Tracking (PRT) is one of two application modules that make up GE Fanuc's CIMPLICITY HMI Tracker product option. This application module is fully integrated with CIMPLICITY HMI software's Base System functionality to enhance its already powerful monitoring capability in a full range of computer-integrated manufacturing environments.

The Base System functionality—Point Management, Alarm Management, Database Logging facilities, and a full-functioned User Interface—enables you to collect data for reporting and to view data in lists, graphic status displays, and alarms. The Tracker module also has the ability to integrate with many of the industry-standard data communications products.

The Tracker Module is a flexible factory floor tool that provides:

- Downtime reporting.
- Production reporting.
- Records of production counts at work stations.
- Graphical monitoring of factory status.
- Fault reporting through direct point values and alarms.

CIMPLICITY HMI's flexible system architecture and modular design allows you to add on options for enhanced functionality. Adding the Production Tracking module to your base system will allow you to track items as they move through your production facility. It will also generate alarms to notify you of tracking exception conditions.

Production Tracking has three points of operator interaction as follows:

- The PRT User Interface for monitoring and controlling your application.
- The Tracker Configuration User Interface for configuring your tracking model data.
- Optional application subroutines for customizing your application.

You can use Production Tracking to:

- Receive information on items as they move through the production facility.
- Request information on the contents of regions through which items are being tracked.
- Add new items to be tracked.
- Delete or modify items currently being tracked.
- Locate specific items being tracked.
- Place or remove *external holds* on tracked items.
- Move items between regions.

# Production Tracking Concepts

Production Tracking (PRT) tracks part/assemblies through the production process using a specialized database to support the Tracking Model.

Within the PRT database:

- Each product is assigned a tracking item that represents its location and associated attributes (for example, color, part number, entry time, etc.).
- Physical areas of the factory bounded by signal devices are assigned regions.
- A product's physical location corresponds to its tracking item's location, defined within PRT by its region and region location.

Products and areas are often associated in some physical or logical fashion on the factory floor, a concept represented within PRT by a hierarchical grouping scheme. PRT allows for one or more items to exist in a single Region Location. These items can be associated with one another to represent the progress of a group of items through the factory.

In turn, each Region can contain one or more of these locations, representing, for example, bins, conveyors and workstations on the factory floor that are in physical or logical proximity to one another. At the next level in the hierarchy is the Group that can hold one or more Regions based on physical location or function.

At the highest level of the hierarchy, it is possible to have multiple copies of the PRT process running simultaneously, each with a unique object name, called PRT Services. Each Service can support an independent list of one or more Groups and Regions. This grouping scheme allows you to expediently search for items or groups of items based on multiple criteria.

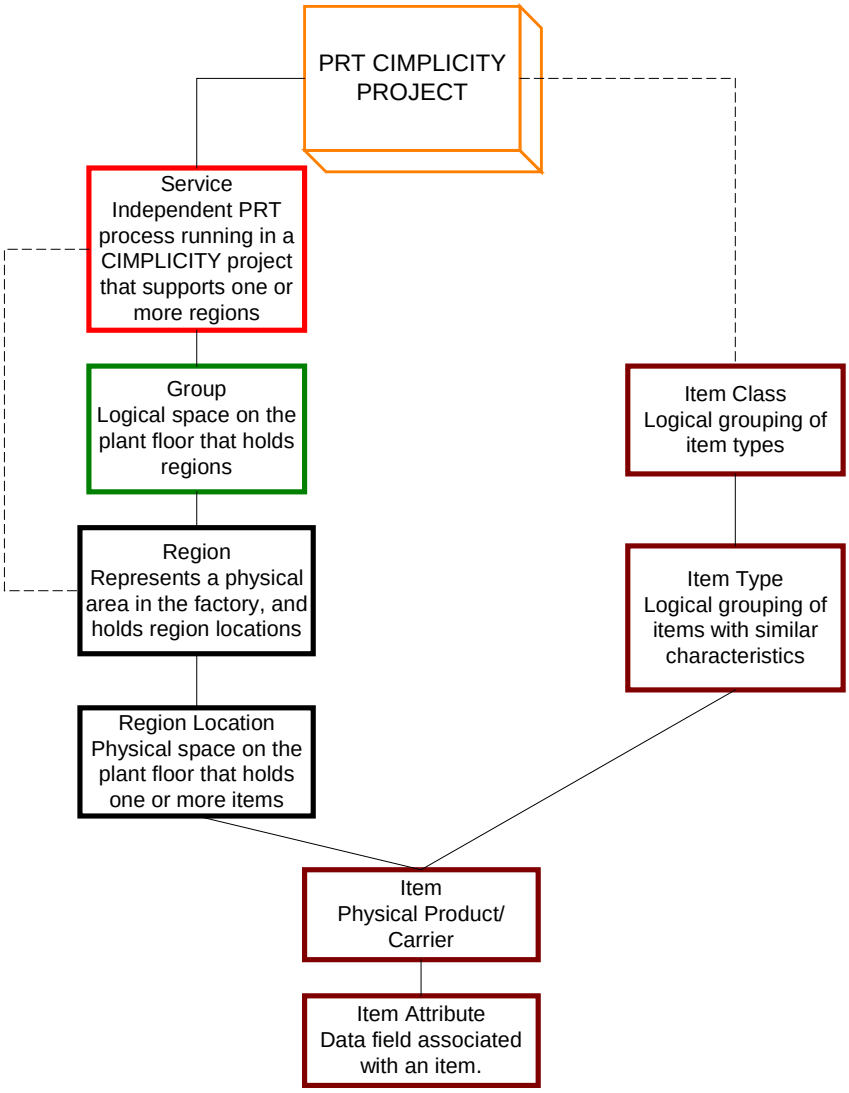
## *Tracker Subsystems*

Tracking Items can be Added, Deleted, Moved, Advanced, or Modified within the PRT database using one of the available subsystems; all of which are listed below.

| Subsystem | Description |
|---|---|
| Tracker Configuration User Interface | Enables configuration and maintenance of Tracking Model data in a graphical user interface. |
| Transition Points | Moves items from region to region based upon location or identification using CIMPLICITY points. |
| Basic Control Engine (BCE) | Write scripts to enable you to perform all of the functions available in PRT. |
| Application Programming Interface (API) | Performs all functions available in PRT. |
| Graphical User Interface (UI) | Performs all functions available in PRT. |
| Production Decision Control (PDC) | Allows complex routing strategies to be developed to control functions of the physical factory using this API-based system. |
| Configurable Tracking Interface (CTI) | Allows items to be added, modified, and moved within the PRT Database using only CIMPLICITY points. |

## *PRT Database Hierarchy*

The PRT database supports the Tracking Model and houses all of the tracking and configuration data. The following diagram illustrates the hierarchical group scheme represented within PRT:


PRT CIMPLICITY PROJECT
Service
Independent PRT process running in a CIMPLICITY project that supports one or more regions
Group
Logical space on the plant floor that holds regions
Region
Represents a physical area in the factory, and holds region locations
Region Location
Physical space on the plant floor that holds one or more items
Item Class
Logical grouping of item types
Item Type
Logical grouping of items with similar characteristics
Item
Physical Product/ Carrier
Item Attribute
Data field associated with an item.

# Tracker Server and Display Options

When you install the Tracker module on a Server or Development Viewer, the two following options are available.

| Option | Installs the |
|---|---|
| Tracker | Entire Tracker option, including the Tracker Display. |
| Tracker Display | Tracker Display only. |

**Note:** The Tracker Display only is available for the Runtime Viewer.

The Tracker Option enables you to:

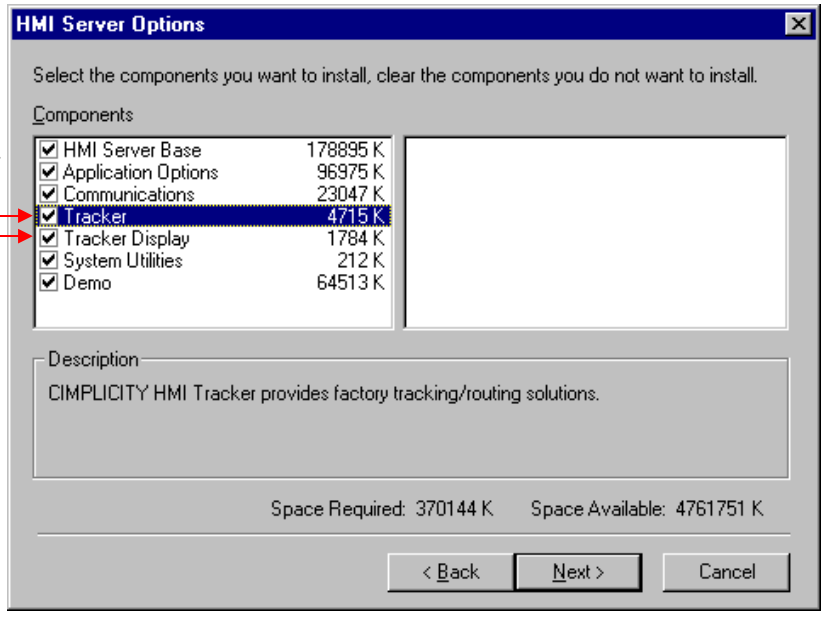
- Access to the Tracker Configuration User Interface.
- Make changes to the PRT database.
- Use all of the functions available in the PRT User Interface.
- View runtime data via CimView screens and scripts.
- Activate alarming.

The Tracker Display Option enables you to:

- Use all of the functions available in the PRT User Interface.
- View runtime data via CimView screens and scripts.
- Respond to alarm conditions.

Provides complete functionality, PRT file configuration, changes to the PRT database and Tracker Display.

Provides access to the PRT User Interface.


HMI Server Options
Select the components you want to install, clear the components you do not want to install.
Components
HMI Server Base 178895 K
Application Options 96975 K
Communications 23047 K
Tracker 4715 K
Tracker Display 1784 K
System Utilities 212 K
Demo 64513 K
Description
CIMPLICITY HMI Tracker provides factory tracking/routing solutions.
Space Required: 370144 K
Space Available: 4761751 K
< Back
Next >
Cancel

# Tracker Installation

Installing the Tracker option on your HMI Server or HMI Viewer takes just moments using your original installation CD for CIMPLICITY HMI.

*See the CIMPLICITY Getting Started manual (GFK-1500) for details about installing CIMPLICITY.*

**To install the Tracker option:**

1. Begin CIMPLICITY installation.
2. Choose to either:
   A. Install the entire CIMPLICITY program or
   B. Add the Tracker option to the same version of an installed CIMPLICITY program.
3. Select the product you are installing as follows:
   A. Proceed to the CIMPLICITY HMI Product Options screen.
   B. Click one of the buttons as follows.

| Button | Installs |
|---|---|
| HMI Server | HMI Server system. |
| Viewer | Either the Viewer Runtime or Viewer Development system. |

   C. Click **Next**.

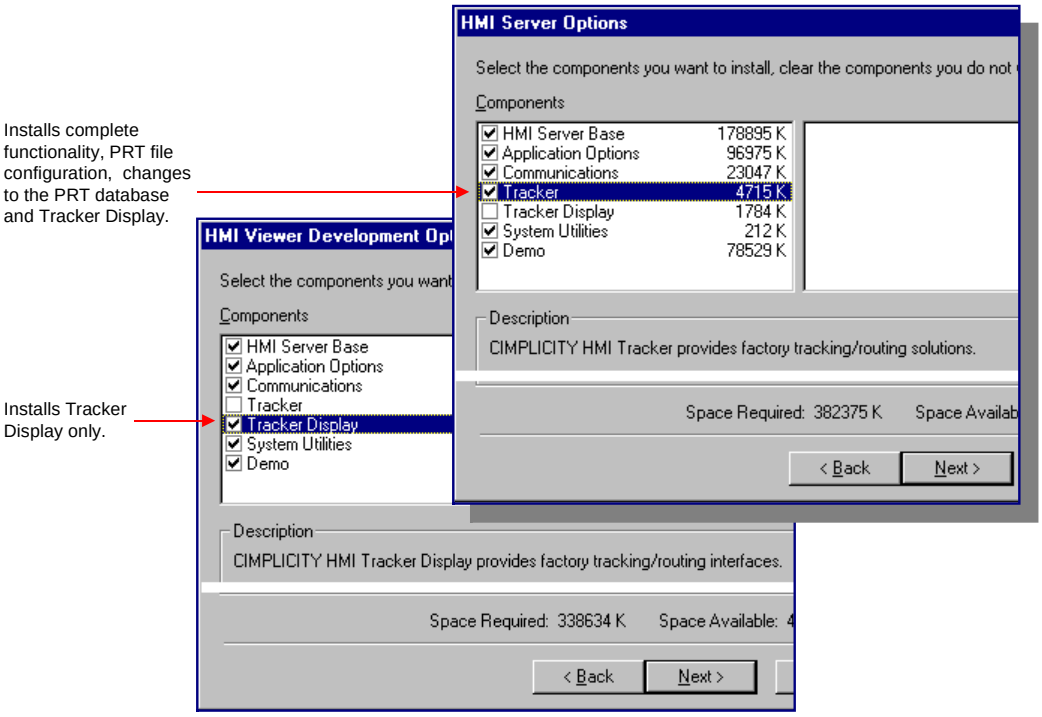
   The screen that displays next depends on which button you click.

| Button | Next Screen |
|---|---|
| HMI Server | The Choose Destination Location screen. |
| Viewer | CIMPLICITY HMI Setup<br>▪ Check either Viewer Runtime or Viewer Development.<br>The Choose Destination Location screen displays. |

4. Proceed to the Options dialog box that displays for the option you chose.
5. Select the following for:

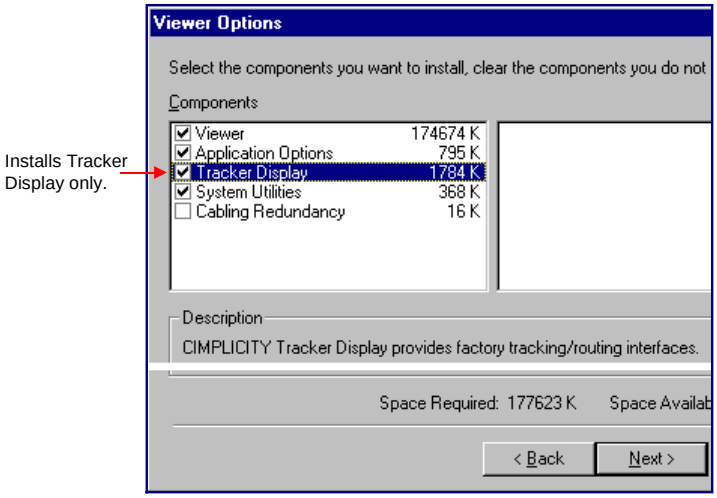
   *Option 1–CIMPLICITY Server or HMI Development Viewer Options Dialog Box*

   A. Check Tracker to install the entire Tracker option including the Tracker Display or
   B. Check Tracker Display to install the Tracker Display only.




*Option 2–HMI Runtime Viewer Options Dialog Box*

Check Tracker Display to install the Tracker Display.


Installs Tracker Display only.
Viewer Options
Select the components you want to install, clear the components you do not
Components
Viewer 174674 K
Application Options 795 K
Tracker Display 1784 K
System Utilities 368 K
Cabling Redundancy 16 K
Description
CIMPLICITY Tracker Display provides factory tracking/routing interfaces.
Space Required: 177623 K
Space Availab
< Back
Next >

6. Click **Next** to proceed with installation.
7. Reboot your computer when the entire CIMPLICITY program (or the Tracker option addition) has been installed.

# Tracker Option Registration

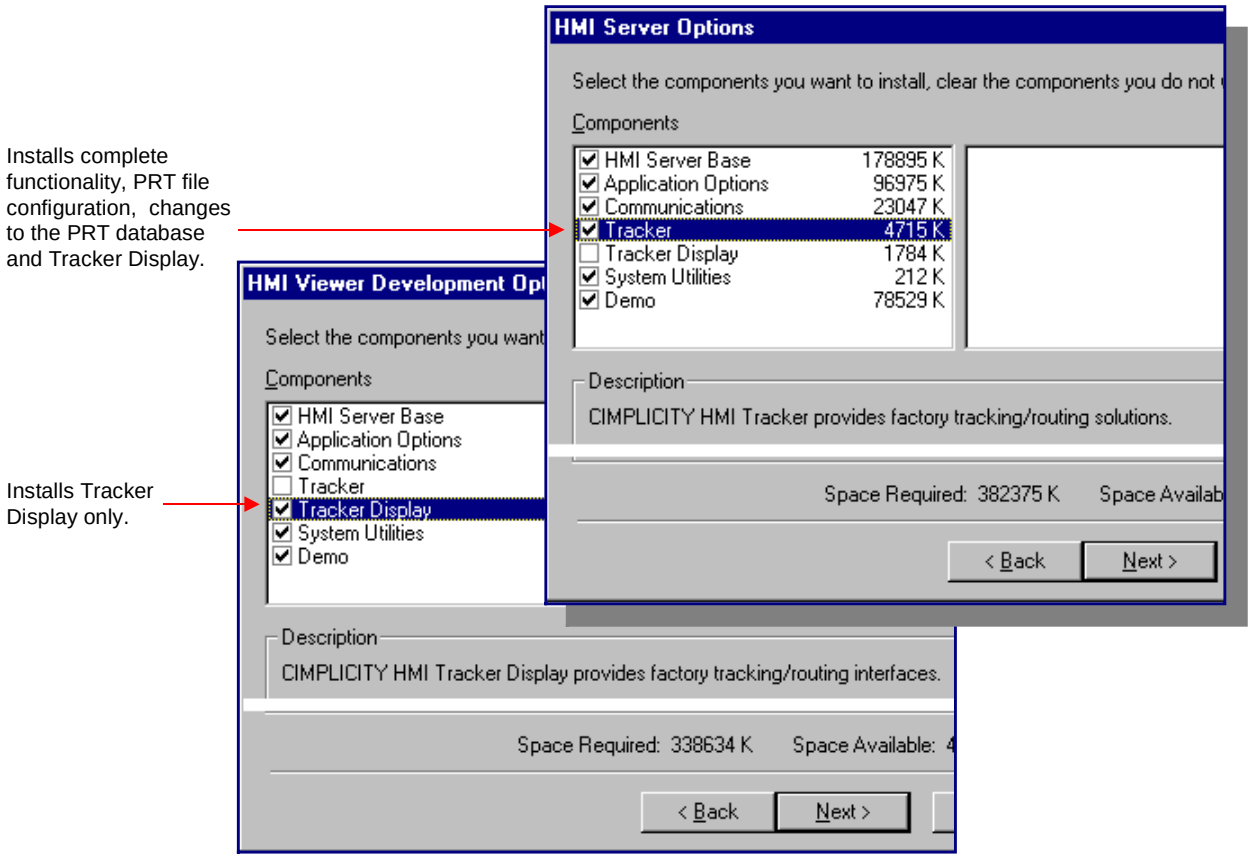
In order to acquire all of the benefits from the Tracker options, register your product using the CIMPLICITY HMI Registration program. Registering your new option is quick and easy as described below.

### To register the Tracker option:

1. Click the **Start** button on the Windows task bar.
2. Select Programs>CIMPLICITY>HMI> Registration

   The CIMPLICITY Registration program begins.
3. Enter the serial number for each Tracker option.

***Result: Tracker options are now registered, and software is fully functional.***



# Production Tracking Overview

## About Production Tracking

Production Tracking (PRT) monitors the progress of Items as they move through a factory production process. Information about an item's location and status is collected as it enters and exits the Tracking Regions. PRT analyzes and updates this information, maintains a log of each item's tracking data, and generates CIMPLICITY alarms to give notification of tracking exception conditions.

Tracking Items are identified by:

- Item IDs.
- Automatic Item-type detectors.
- Manual transactions.
- Application processes interfacing with PRT.

## Using the Tracking Model Summary

The PRT module uses a Tracking Model that you design and configure to support its tracking functions. The PRT Tracking Model is configured using the Tracker Configuration User Interface. The Tracker Configuration User Interface provides a directory structure for maintaining data and associated dialog boxes for configuring PRT data.

The PRT tracking model defines:

- Tracking regions.
- Entry and exit transition indicators (bounding Regions).
- Item types.
- Tracking events to log.
- Exception conditions for generating alarms.
- Points required to support the graphic display of Tracking data.

The Tracking Model can be configured to track:

- Individual items.
- Items combining to reside at a single location (Combine Items Tracking Region).
- Items dispersing to reside at individual physical locations (Disperse Items Tracking Region).
- Associated items (both physically and logically connected).

**Important:** A source PRT project must have a remote project login configured for the destination project through the prt_client.

*See the Base System User's Manual, GFK-1180, the System Management chapter, Remote Projects section" for details about configuring for remote projects.*

## Tracking Regions

A ***Tracking Region*** is bounded by devices that inform the tracking model that an item has entered or exited the region. The Region's entry and exit indicators are responsible for providing an Item's location data. Except for 'production start' and 'production stop' Tracking Regions, the exit transition indicator for one Region must always be configured to be the entry transition indicator for the destination Region in the production line. An Item tracked by PRT must always reside in a Tracking Region. Each tracking region must be associated with a primary and a secondary type.

The following Region types are supported:

| Primary Region Types | Secondary Region Types |
|---|---|
| Pool Tracking | Normal Tracking |
| Sequential Tracking | Combine Items Tracking |
| Shifting Tracking | Disperse Items Tracking |

### *Primary Region TypesSecondary Region TypesPool Tracking*

A Pooled Tracking Region does not process Items in any particular order. The first item into the region is not necessarily the first item out. Items must be positively identified in order to be tracked through a Pool Tracking Region.

Configured Tracking Regions may or may not map directly to physical locations on the factory floor. If a void exists between two configured Regions in the factory, a Pool Tracking Region must be configured to exist between those two Regions.

**Note:** Regions of tracking type, DETAINMENT are treated as Pool Regions.

### *Sequential Tracking*

A Sequential Tracking Region maintains the relative sequence of the Items within the Region. Therefore, the first Item into the Region must be the first Item out; otherwise, a tracking exception condition occurs generating an alarm.

### *Shifting Region Tracking*

Shifting Regions support empty locations intermixed with occupied locations. As Items are transitioned into a shifting region, all Items currently in the region are shifted by one location, and the Item at the head location is transitioned to the Shifting Transition Region. Items are always transitioned to the end location of the region, not the last occupied location.

*For more information see "Appendix C: Shifting Regions."*

### *Normal Tracking*

A Normal Tracking Region tracks the flow of Items or a group of Items. Items can be serialized or non-serialized.

### *Combine Items Tracking*

A Combine Items Tracking Region allows multiple items to converge into one region location. There is only one associated region location associated in the Combine Items Region.

### *Disperse Items Tracking*

A Disperse Items Tracking Regions is the opposite of a Combine Items Tracking Region, in that combined items are separated into successive regions. There is only one region location associated with the Disperse Items Tracking Region.

## Associating Items

Production Tracking permits Item associations to be formed:

- Automatically in a Combine Items Tracking Region
- By request through a manual transaction or an application interface message.

### *Automatic Item Association*

Automatic Item associations occur in Combine Items Tracking Regions where configuration data specifies the Item types that will participate in an Item association and the role each Item will play in the relationship. Each time an Item enters a Combine Items Tracking Region, where automatic Item associations occur, the Item type is checked to determine if this Item participates in an Item association. An automatic Item association takes place when each of the Items in the association have arrived in the Combine Items Tracking Region.

### *Parent-Child Item Association*

A Parent-Child Item Association requires two Items: The first Item must be serialized and is the parent Item in the relationship. The second Item must be non-serialized and is the child Item in the relationship. Once associated, the parent and child Items remain logically connected in the production process. Tracker allows the parent Item may have more than one associated child Item.

A parent Item and a child Item can be associated by request, whether they reside in the same tracking region or in separate regions.

The request must specify the:

- Sequential tracking region in which the child Item resides.
- Item type of the child Item (non-serialized).
- Sequence location of the child Item in the region.

## Modifying an Item's Status

A Tracking Region can be configured to affect an Item's status due to Item-hold specifications. For example, a shortage of an Item attribute occurs if a required physical Item attribute (for example, A/C or seat style) required is shown as being in short supply in a Region on the production line.

An Item's status can be modified by:

- An external process.
- A manual transaction.

An external process:

- Can modify an Item's status to Hold when shortage attributes exist.
- Is configured to provide Item-hold information for Serialized Items.
- Enables PRT to modify the status of all Items residing in a Tracking Group.
- Modifies the status to Hold for each new Item entering the Tracking Group.
- Can be configured to generate a CIMPLICITY alarm when an Item's status is modified to Hold.
- Can deactivate Item-hold status on all affected Items in a Tracking Group.
- Will not impact a Hold placed by a manual transaction.
- Monitors the Item's status; and, if capable, physically holds the item at a pre-configured point on the production line.

A manual transaction:

- Can be executed to modify an Item's status to Hold.
- Can have comments attached to describe the reason for the Hold.
- Is held separately from a Hold placed by an external process.
- Can deactivate Item-hold status on all affected items.
- Will not impact a Hold placed by an external process.

## Monitoring Sequences

A sequence error may be detected at the exit transition indicator of a Sequential Tracking Region when an Item does not arrive at the transition indicator in the expected order. You will configure the action to be taken in a Tracking Region when an Item sequence error is detected.

The five configuration options are to:

- Immediately place the item in the corresponding detainment Region.
- Change the Item's status to DELAYED. Once the configured number of Items has exited the Region, the missing Item is placed in the corresponding detainment Region.
- Scrap the missing Item.
- Change the Item's status to DELAYED. Once the configured number of Items has exited the Region, the missing Item is scrapped.
- Do nothing—sequence monitoring disabled.

Synchronization

In the event a sequence error is detected, PRT allows resynchronization of the tracking environment, which can be done either manually or automatically. Further, if Automated Region Validation is configured, sequence processing occurs when the Items in the Region are reordered.

You will learn about:

### *Manual Resynchronization*

The tracking environment can be manually resynchronized by reconciling the Tracking Region queues using available functions in the PRT User Interface.

Available functions are as follows:

- Add Item
- Delete Item
- Move Item
- Fetch Item
- Advance Item
- Reorder Region

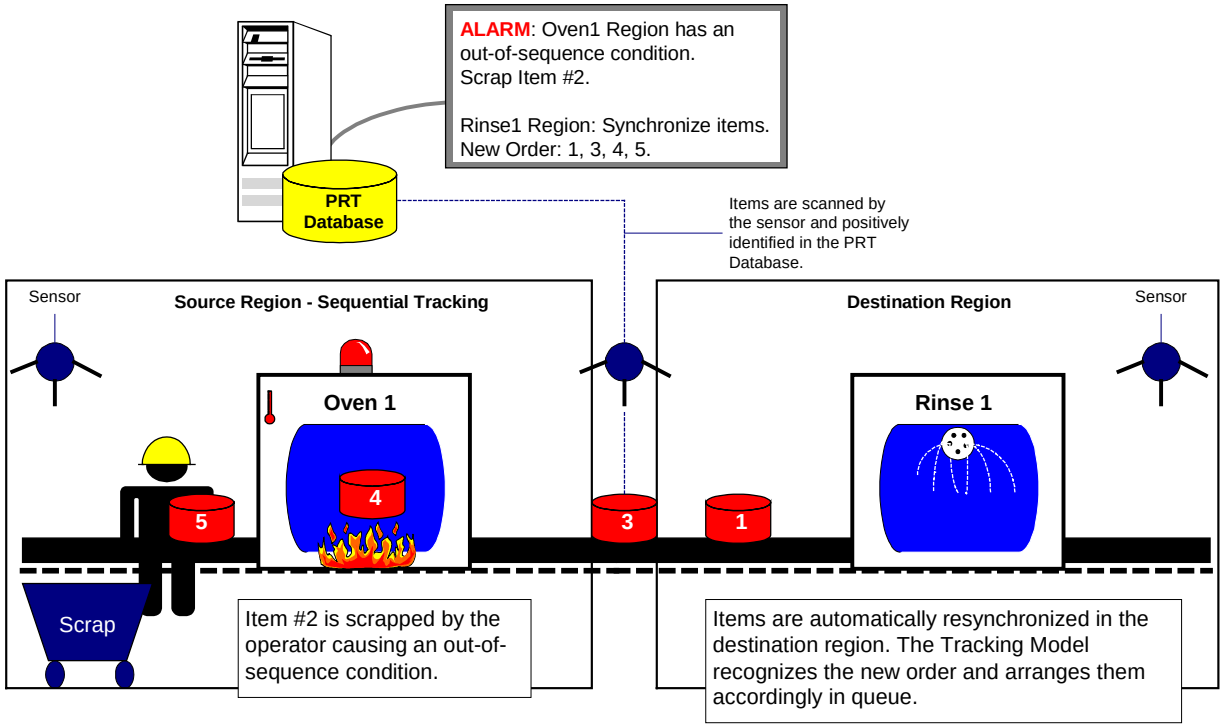
*For additional information about these functions refer to "Using the Production Tracking Window."*

## *Automatic Resynchronization*

Automatic resynchronization takes place when a serialized Item exits a Sequential Tracking Region in an out-of-sequence condition and is then resynchronized in the destination region. PRT will process the Item according to the configuration for detected sequence errors in the Region. See figure.

PRT will:

- Ensure the serialized Item is correctly identified in the queue of the destination region.
- Assume the sequence of the Items in the source region is incorrect until the missing Item is scrapped or the delayed Item is removed.
- Identify the item in the destination region when it is removed from the detainment region and placed in the destination region.


ALARM: Oven1 Region has an out-of-sequence condition.
Scrap Item #2.
Rinse1 Region: Synchronize items.
New Order: 1, 3, 4, 5.
PRT Database
Items are scanned by the sensor and positively identified in the PRT Database.
Sensor
Source Region - Sequential Tracking
Destination Region
Sensor
Oven 1
Rinse 1
4
5
3
1
Scrap
Item #2 is scrapped by the operator causing an out-of-sequence condition.
Items are automatically resynchronized in the destination region. The Tracking Model recognizes the new order and arranges them accordingly in queue.

**Note:** Regions configured with Automated Validation will potentially resynchronize every Item as they enter the head location. *For additional information see "Automated Region Validation."*

### *Items Scrapped*

Production Tracking allows Items to be scrapped through a manual transaction or an application interface message. You can configure your Tracking Model to log the scrap event.

PRT logs:

- The Item ID and Type for serialized Items
- The Item Type for non-serialized Items.
- Any comments entered for description purposes.

## Generating CIMPLICITY Alarms

Production Tracking generates CIMPLICITY alarms when exception conditions are detected during Item tracking. You will configure when alarms are to be generated on a region-by-region basis. For example, an alarm is generated if a job transitions and does not follow its predefined route.

## Configuring CIMPLICITY Setpoints

CIMPLICITY Setpoints can be configured to enhance your Tracking Model. There are numerous uses for Setpoints as described below.

You can configure Setpoints to:

- Specify the status of the Region.
- Indicate the quantity of Items in the Region.
- Calculate the quantity of Items in the Region by Item Type.
- Update when an Item is automatically moved into a Region (Point-based move).
- Inform when an Item passes a Region transition indicator, to include the Item ID for serialized Items and the Item type for non-serialized Items.

## Using the PRT User Interface

Production Tracking comes complete with a full-function user interface to display and manage tracking data for your Regions and Items.

With the PRT User Interface you can:

- View all the Tracking Regions in a Group.
- Display properties for an Item.
- Scrap Items.
- Move an Item to another region.
- Add or delete Items.
- Reorder Items or Item carriers.
- Advance Items in the Tracking Model.
- Connect to multiple projects.
- Perform all of these functions and much more!

## Tracking Items

Production Tracking begins tracking an Item when a production start message is received and stops tracking an Item when it exits the last Region on the production line. Serialized Items require an Item ID, and non-serialized Items require an Item type ID in order to be tracked through the system.

PRT can begin tracking an Item:

- Through a manual transaction.
- Through a message from a PRT application interface.
- By a CIMPLICITY Point.

### *Non-Serialized Items*

PRT provides the reserved Item type $UNKNOWN, which can be associated with a non-serialized item at the start of production.

There are certain guidelines that must be followed when using this association as follows:

- The association can only take place when a non-serialized Item passes a transition indicator.
- The transition indicator must be configured as an Item type detector.
- A CIMPLICITY Point must be configured to report positive identification of the Item type.
- An application interface message can associate a non-serialized Item with an unknown Item type.
- The Item type of the non-serialized item must be identified later in the production process.

### *Serialized Items*

When a serialized Item is started through the production process, PRT requests the Item type, status and attribute data. Once the Item is started through the production process, tracking data is then collected and logged. PRT will process Items differently depending on the method used to start production.

If production start is originated by a:

- CIMPLICITY Point, configuration data determines if Production Tracking will request an external process for additional information.
- Manual transaction or an application interface message, additional tracking data may be included in the message. Also, an external process may be queried for additional information.

The following guidelines apply to serialized Items:

- Each serialized item is assigned a unique Item ID.
- They can also be identified and tracked by a reference ID.
- A reference ID acts an alternate key to locate the Item and Item ID.
- The reference and Item IDs can be automatically associated through configuration data.
- Reference and Item IDs can only be associated when an Item is positively identified by a transition indicator.
- A CIMPLICITY Point must be configured to report the reference ID through the transition indicator.
- Reference and Item IDs can also be associated through an application interface message.

## Using Item Classes

Item classes are a way of grouping together item types in order to further simplify and expedite searches in the PRT database.

Item Classes:

- Group item types together to create a larger subset in the database.
- Are independent of region, group or service.

**Tip:** Although item types do not have to belong to an item class, it is strongly recommended that you create item classes. Feel free to create just one item class to which all of your item types can belong.

## Generating Reports

Production Tracking provides the capability to generate reports using logged data. A common method for logging data is by using Microsoft's ODBC SQL Server, and then generating reports using a report writer.

## Using the PRT Interface to External Software

The Production Tracking process provides Tracking Region data and Item tracking data to external application programs through the PRT Application Interface. PRT also accepts requests through the application interface to dynamically provide interested external application programs with Item tracking data modifications. The external process must send PRT a message to indicate it is requesting to become an interested process in order to receive dynamic serialized Item updates.

## Tracking Data

You can export and import tracking data to and from the PRT process using:

- Export Tracking Data.
- Import Tracking Data.

### *Export Tracking Data*

The EXPORT TRACKING DATA function provides a mechanism to export the current tracking data for a single PRT process to a file. This function can be used if static configuration data must be modified (for example, PRT is shutdown to add or delete a tracking Region). The current Tracking Region and Item tracking data is exported to create a file that can be edited to match the modified static configuration data.

### *Import Tracking Data*

Using the IMPORT TRACKING DATA function, the edited files can be reloaded into the tracking queues without manual entry of the data for each queue.

The IMPORT TRACKING DATA function copies exported Tracking Region and Item tracking data from the export file to the tracking queues configured for the current PRT process. The import and export tracking data functions can be used to modify the tracking queues for a PRT process without shutting it down; that is, as long as PRT static configuration data is not modified between the export of the file and the import of the edited file.

# CIMPLICITY Software Component Interface Requirements

Production Tracking interfaces with a number of systems to provide increased functionality.

PRT interfaces with:

- CIMPLICITY HMI Base System Point Management
- CIMPLICITY HMI Base System Alarm Management
- CIMPLICITY HMI Application Processes
- PRT User Interface.

## *Point Management Process to PRT Process*

PRT interfaces with CIMPLICITY HMI's Point Management module in order to receive CIMPLICITY Point data values. PRT is capable of receiving Point data from multiple Point Management processes. Configuration data specifies the information Production Tracking will provide to Point Management.

The interface with Point Management:

- Transfers information on Items entering and exiting Tracking Regions
- Provides Point Management with information on Tracking Region status
- Calculates the quantity of Items in a Tracking Region and the quantity of Items by type in a Tracking Region.

## *PRT Process to Alarm Management Process*

PRT interfaces with Alarm Management for notification of alarm conditions. PRT can be configured to generate alarms when exception conditions are encountered as Items are tracked. Users are notified of CIMPLICITY HMI alarms based on their assigned role and their view of configured Resources.

## *Application Process to PRT Process*

PRT accepts unsolicited messages from application processes requesting Item and Tracking Region data. Messages are also accepted requesting Item tracking data and Tracking Region data modifications. Application processes can send messages to PRT to initiate production start for serialized and non-serialized Items. Application processes will communicate to PRT through a limited set of PRT application functions used to send the requests.

## *PRT User Interface*

Your tracking system can be viewed graphically using the CIMPLICITY CimView module. Custom displays and scripts can be created to show a detailed view of your production process. Also provided with Tracker is the PRT User Interface, which provides a view of all the regions and corresponding data. You can perform a wide variety of functions from the User Interface giving you increased control over your tracking model.

# Using the Production Tracking Window

## About the Production Tracking Window

The Production Tracking Window (user interface) is a runtime window that enables you to view and manage production tracking data. Using the menu options and toolbar you can perform a variety of functions that allow you to control production flow on the factory floor.

The PRT_UI is designed to simplify management of your PRT data with an easy to use graphical interface. The PRT_UI is displayed on a Windows platform enabling flexibility in performing functions through menus, toolbar buttons, quick keys and popup menus.

You can do all of the following from the PRT_UI:

- Display and browse for regions.
- Connect to multiple projects.
- Move items between projects.
- Add, insert, delete, fetch and advance items.
- Find items.
- Reorder items.
- And much, much more!

# Procedure to Open the PRT Window

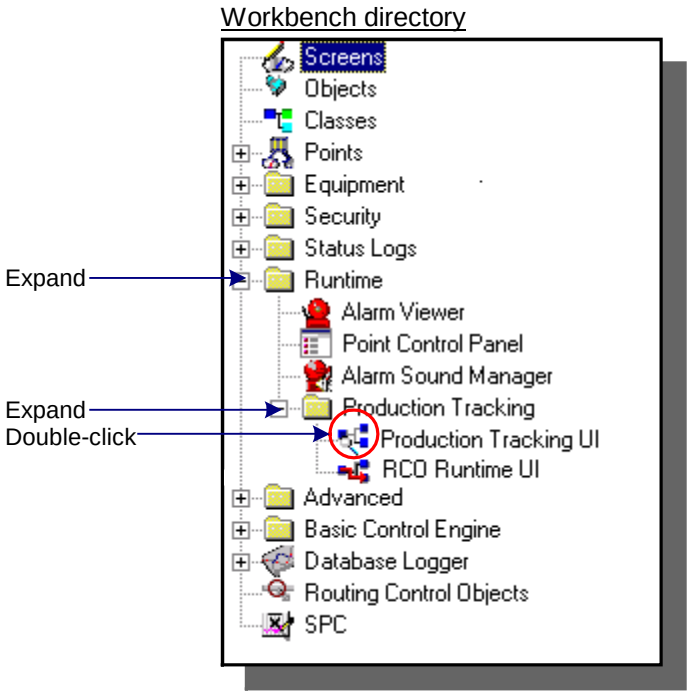
**To open the PRT window:**

*Method 1*

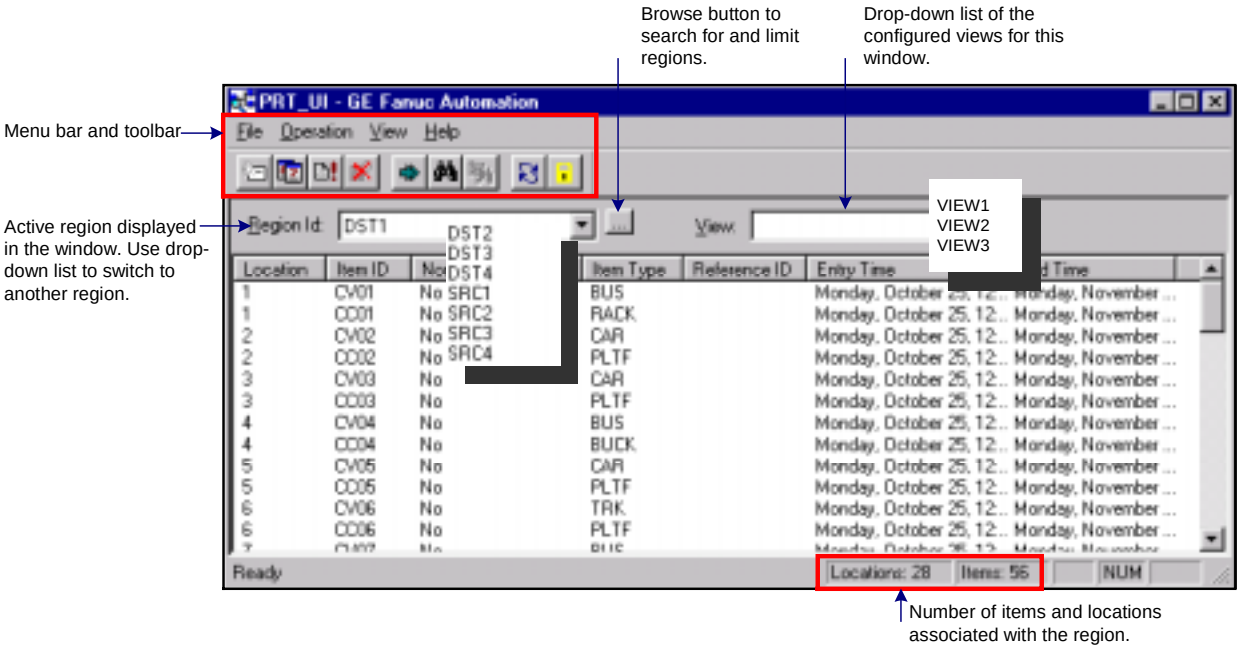
1. Run your Tracker project.
2. Click **Start** on the Windows task bar.
3. Select Programs>CIMPLICITY>HMI>PRT User Interface.

*Method 2*

1. Run your Tracker project.
2. In the Workbench directory, expand the Runtime folder.
3. Expand the Production Tracking folder.
4. Double-click the Production Tracking UI icon.



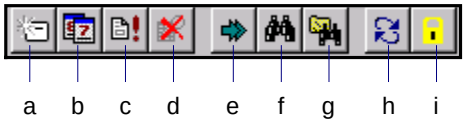

***Result: Using either method, the PRT_UI window opens. The Region Id field indicates the active region in the window, and the item count is displayed on the status bar.***




**Note:** The columns you see in this window are configurable and are explained in the *"Configuration Requirements"* chapter.

## *PRT Toolbar*

The buttons on the PRT toolbar provide one-click access to the most commonly used functions. They are described as follows:




**a.** Adds a new item location to the selected item.
**b.** Inserts a new item "in front" of the selected item.
**c.** Modifies the selected item's properties.
**d.** Deletes selected item from inventory.
**e.** Moves selected item from current region to another region.
**f.** Finds region where item is located.
**g.** Searches PRT database for an item.
**h.** Refreshes PRT DV regions.
**i.** Auto locks the status bits in the region, when configured as such.

# Procedure to Set Information Parameters

When you initially access the Production Tracking window (PRT User Interface), all of the regions in the current project are displayed by default. You have the option of setting parameters to customize the information you want to display when you open the PRT window. Start by placing a shortcut for the PRT_UI on your Windows desktop.

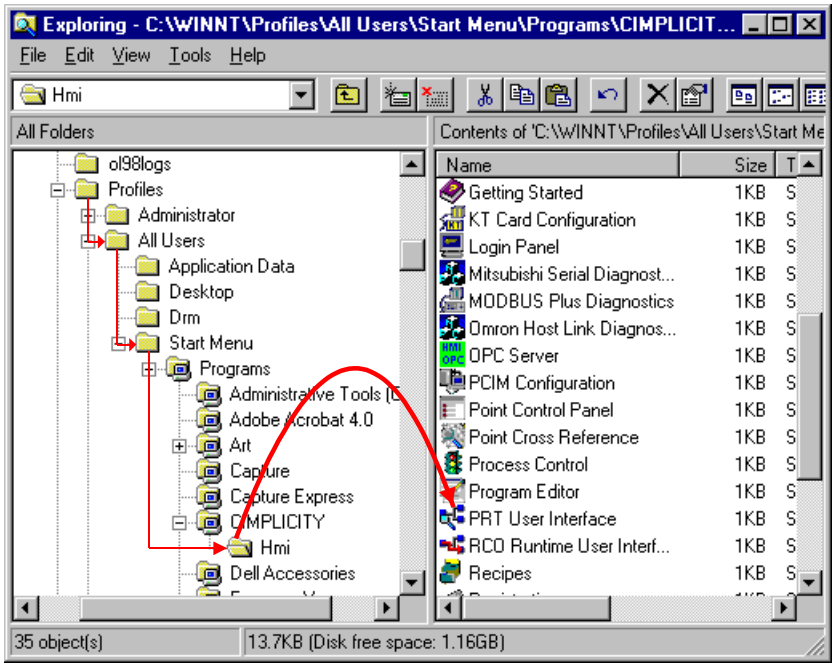
Do the following:

- Create a shortcut.
- Set parameters.

## *Procedure to Create a Shortcut for PRT User Interface*

**To create a shortcut for the PRT User Interface:**

1. Open Windows Explorer.
2. Expand the Windows folder (e.g. WINNT).
2. Navigate to the Profiles>All Users>Start Menu>CIMPLICITY HMI workgroup directory.




3. Drag the PRT User Interface shortcut icon onto your Windows desktop while holding down the right-mouse button.

   A popup menu displays when you release the right-mouse button.

5. Select Create Shortcut(s) Here.

***Result: The PRT User Interface shortcut displays on your Windows desktop.***

## *Parameters Set for the PRT User Interface*

**To set parameters for the PRT User interface:**

1. Right-click the PRT User Interface shortcut on the Windows desktop.

   A popup menu displays.

2. Select Properties.

   The PRT User Interface Properties dialog box displays.

3. Select the Shortcut tab.

3. Enter the switch and parameter to customize the PRT User Interface when opened In the Target field. Use the following switch options:

| Switch | Parameter |
|---|---|
| **`-R <region name>`** | where <**region name**>is a valid region name. Displays information for a specific region. |
| **`-S <prtsystem>`** | where <**prtsystem**> is the PRT server to connect to in the project. Enables use of a particular PRT service. If none is specified, all regions on any PRT system are displayed. |
| **`-N <node/project>`** | where <**node/project**> specifies the node or project name with which to connect. Enables connection to the named node or project. If none is specified, the connection is made to the local project. |
| **`-P <path>`** | where <**path**> is the path name where the configuration files are loaded. This option must be used if accessing a remote project. |
| **`-I <file path>`** | where <**file path**> is the name of the file used to connect to multiple projects. |

4. Click **OK**.

***Result: Parameters indicated in the Target field will be triggered when the PRT User Interface is accessed.***

**Note:** If you are viewing the PRT User Interface from a Viewer, you do not have to use the

**–N** switch on the command line to connect to a remote project. When the PRT_UI is initially launched, a dialog box will display allowing connection with a remote project that has been enabled for broadcast.

# Common Navigation Techniques

The PRT window is used to display information about items in a particular region. Items for one region at a time can be viewed in the PRT window. Your production facility can have anywhere from several to many regions and the PRT_UI offers you the flexibility to display other regions and to search for other regions in your database.

The most common navigation techniques are:

- Display another region.
- Browse for regions.
- Modify criteria when browsing for regions.

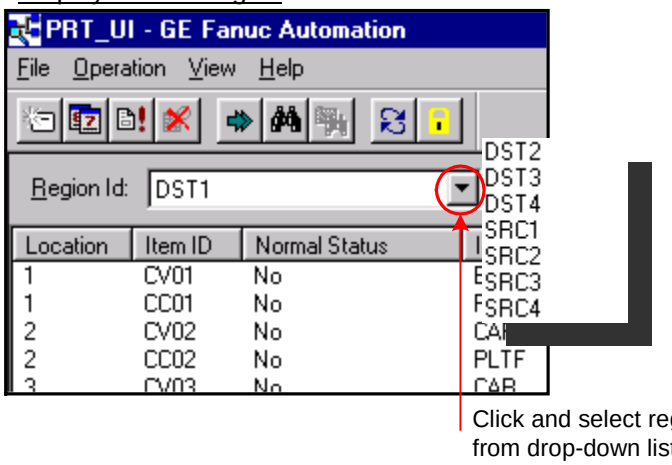
## *Procedure to Display another Region*

You can view additional regions for the local project in the PRT window.

**To display another region:**

1. Click the down arrow in the Region ID field to display the list of configured Region IDs.
2. Select the region from the drop-down list.

Display another region



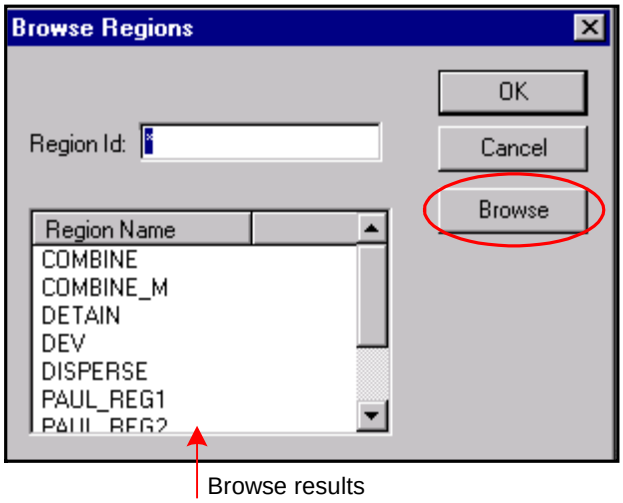

***Result: The window displays the list of items for the selected region and update the item count on the status bar.***

## *Browse Regions*

Use the **Browse** button on the PRT window to select a region from the list to display, or to reduce the number of regions listed by entering criteria to display a select group. From the selected group, you can then choose the region you would like to view.

### To browse for a region:

1. On the PRT window, click ... next to the Region ID field to open the Browse Regions dialog box.




2. Do one of the following to select a region for viewing:
   A. In the Region Id field, type the region name or
   B. Select a region from the list.
3. Click **OK** to view the region in the PRT window.

### *Procedure to Modify Criteria for Browsing Regions*

**To reduce list of regions:**

1. Click ... next to the Region ID field to open the Browse Regions dialog box in the PRT window, .
2. Enter criteria to reduce the list of Region Names in the Region Id field.

   **Tip:** Use the wildcard character (*) to increase search result.

   **Example**

   - *R displays all regions ending in R.
   - S* displays all regions beginning with an S.
   - *T* displays all region with a T in the name.
   - * displays all available regions.
3. Click **Browse** to display the search results.
4. Select a region from the group.
5. Click **OK** to view the region in the PRT window.

# PRT UI Functions

Everything you need to monitor and control your tracking data can be performed from the PRT UI window. The menu options, toolbar buttons and drop-down lists give you increased flexibility in performing everyday tasks.

You can do the following:

- Connect to multiple projects
- Move items between projects
- Add and insert items to a region
- Modify an item in a region
- Delete items in a region
- Fetch an item from one region to another
- Move an item from one region to another
- Advance an item from an exit location to an entrance location
- Find a specific item from within all regions
- Reorder the items in a region
- View and update a region status
- Print reports based on configured files

## Connecting to Multiple Projects

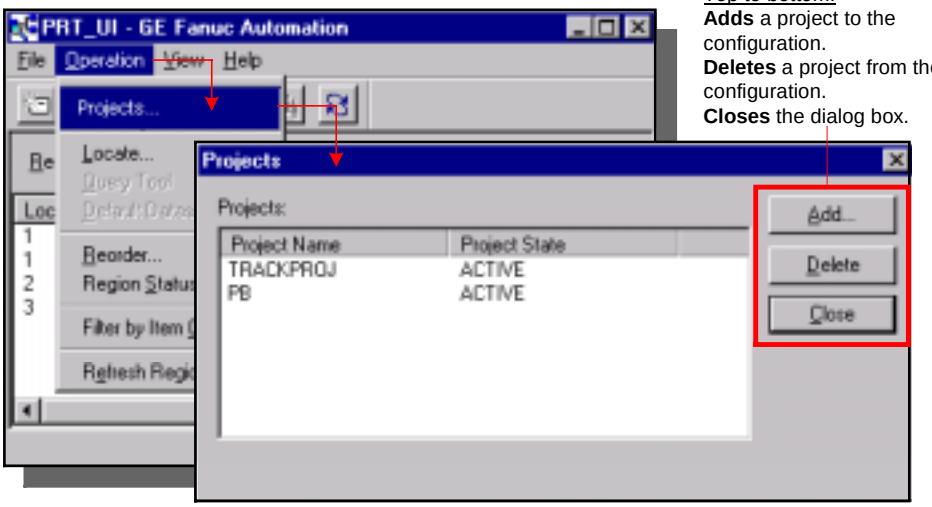
You can view regions in multiple projects without having to open multiple PRT windows. In order to view multiple projects, PRT Services must be running on each project and you must have access privileges to the projects you want to view. Projects can be added or deleted dynamically in the PRT window allowing you to limit or expand the number of regions in view. Also, you can save your changes and access the file when you want a particular view.

### To connect to multiple projects:

1. Click the Operation menu.
2. Select the Projects option.

***Result: The Projects dialog box opens listing all available projects.***

Managing projects




Top to bottom:
**Adds** a project to the configuration.
**Deletes** a project from the configuration.
**Closes** the dialog box.

**Important:** If you want to move items across project boundaries, you have to do additional PRT startup configuration.

*See the appendix "Configuring PRT Client Setup" in this manual for details.*

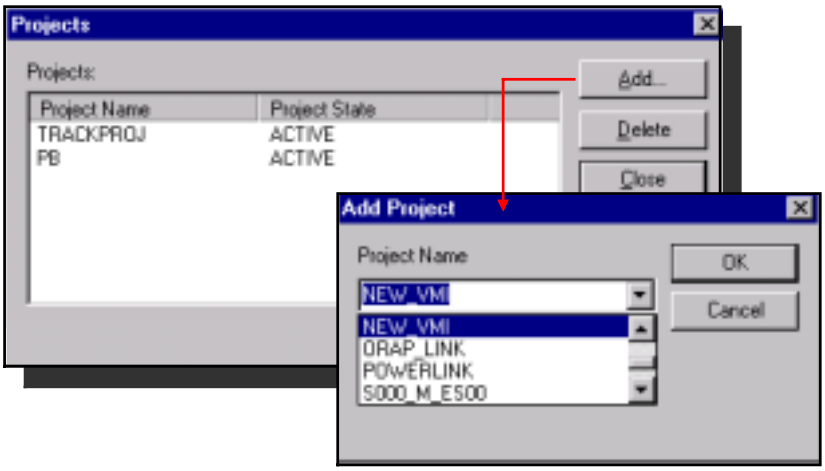
## *Procedure to Add a Project*

**To add a project to the PRT:**

1. Click **Add.**

   The Add Project dialog box opens.

Add a project




2. Select the project you want to add from the Project Name drop-down list.
3. Click **OK** to accept or **Cancel** to cancel the operation.
4. Click **Close** in the Projects dialog box.

   A message box will display asking if you want to save your changes.
5. Click **Yes** to save your changes in a file and proceed to the Save As dialog box.

   **Note:** Click **No** to cancel the operation.
6. Navigate to the appropriate directory in the Save As dialog box.
7. Click **Save** to save your configured file.

***Result: The project is added to your configured file.***

### *Procedure to Delete a Project*

1. Select the project you want to delete from the Project Name drop-down list.
2. Click **Delete**.

   A message box displays asking if you want to delete the project.

   **Important:** If by deleting the project there will be no active projects in the PRT window, a warning message will display stating that the PRT UI will be terminated due to no projects. Clicking **Yes** on this message box will close the PRT window.
3. Click **Yes** to delete the project and to save your changes in a file.

   A Save As dialog box opens.

   **Note:** Click **No** to cancel the operation.
4. Navigate to the appropriate directory in the Save As dialog box.
5. Click **Save** to save your configured file.

***Result: The project is deleted from your configured file.***

**Tip:** You can access the file configured to view multiple projects using the Command Line in the Program Properties dialog box. *See topic, "How to Set Parameters" earlier in this chapter for details.*

## Moving Items between Projects

When you are connected to multiple projects in the PRT window, you can move items between projects. In the PRT window, the active region is displayed in the viewing window and all of the corresponding information is loaded into the appropriate fields. You can move an item from the viewing window into another region within another project. There are, however, some issues to take into consideration, each of which are explained in detail as follows:

- Item status
- Item types

**Note:** Skip ahead to topic, *Move an Item* for instructions on moving items across project boundaries.

#### Item Status

The item status must be identical for items on different projects. The item status is a 32-bit number for all projects; however, if the same bits have different meanings from project to project, problems will arise. Modify the status using the Status Configuration dialog box in the Tracker Configuration UI on all projects to have identical statuses in order to move items effectively.

#### Item Types

In order to move an item between projects, the item type must be allowed in the region of the project receiving the item. The item class will be as it is configured in the PRT Item Type Configuration dialog box. If the item type is not allowed in the recipient project, a warning message will display the move will be disallowed.

## Adding, Inserting or Modifying an Item

In order to add, insert or modify an item, ensure that the active region in the PRT window is the region in which you want to make changes. See earlier instructions for displaying a region. Because the dialog boxes and procedures for adding, inserting and modifying items are so similar, following are initial instructions for opening the individual dialog boxes. The procedure then continues in Step 4.

Perform Step 1, Step 2 or Step 3, then continue with Step 4:

**Step 1** Add an item to a region.

**Step 2** Insert an item in a region.

**Step 3** Modify an item in a region.

**Step 4** Continue procedure from Steps 1, 2 and 3.

**Step 5** Set Attributes for item.

**Step 6** View and modify item status.

## *Step 1. Add an Item to a Region*

**Step 1. Add an item to a region**:

1. Do one of the following:

   *Method 1 – Menu option*

   A. Click File on the menu bar.

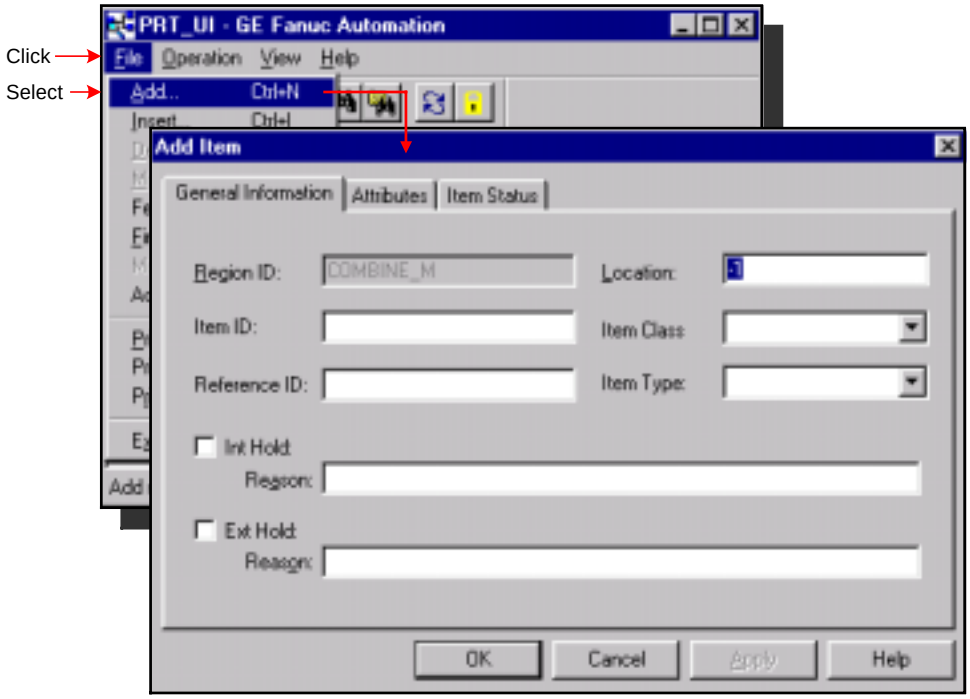
   B. Select the Add option;

   *Method 2 – Quickest*

   Click on the toolbar

   The Add Item dialog box opens when you use either method.

Example using menu option




2. Go to Step 4.

## *Step 2. Insert an Item in a Region*

**Step 2. Insert an item in a region:**

1. Do one of the following:

   *Method 1 – Menu option*

   A. Click Operation on the menu bar.

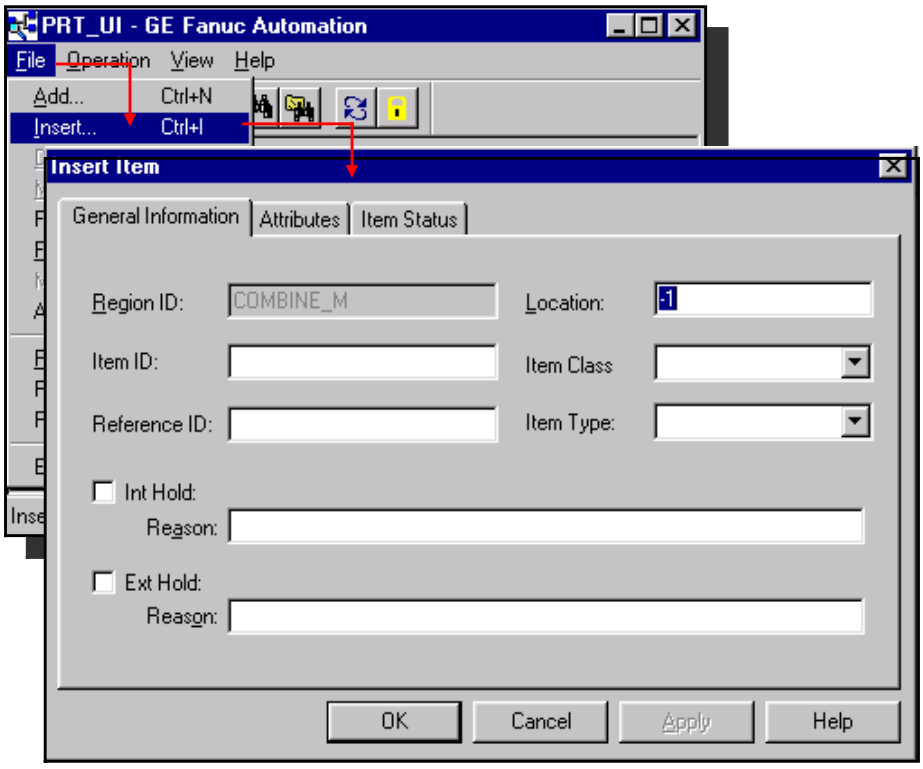
   B. Select the Insert Before Location option.

   *Method 2 – Quickest*

   Click on the toolbar.

   The Insert Item dialog box opens when you use either method.

Insert an item




2. Go to Step 4.

## *Step 3. Modify an Item in a Region*

**Step 3. Modify an item in a region:**

1. Do one of the following:

   *Method 1 – Menu option*

   A. In the PRT window, select the item in the list.

   B. Click Operation on the menu bar.

   C. Select the Modify Item option

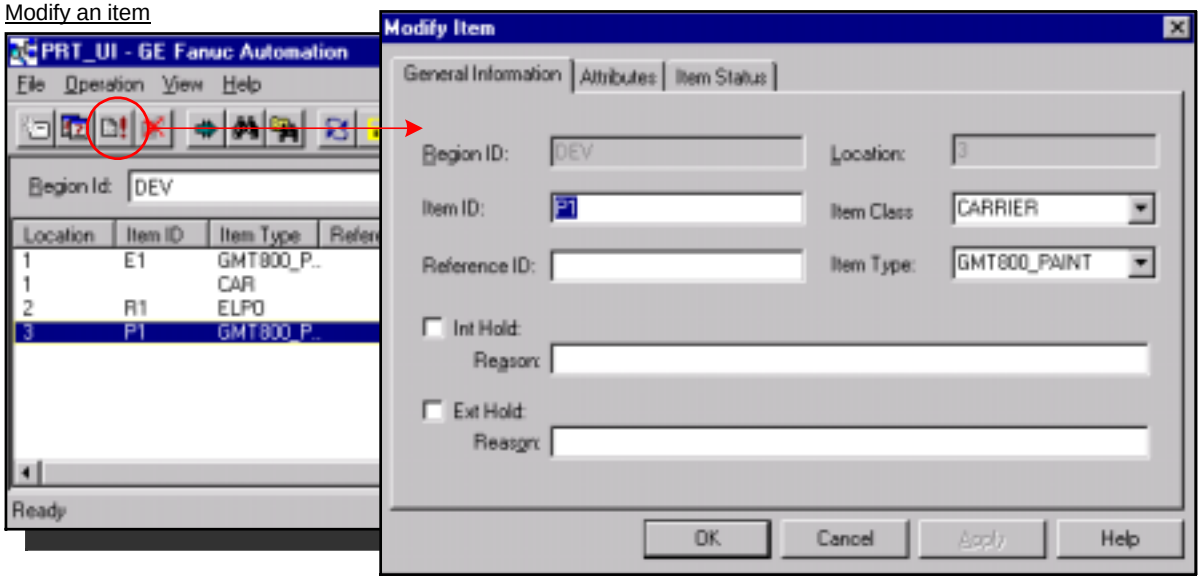
   *Method 2 – Quickest*

   A. Select the item in the list.

   B. Click on the toolbar

   The Modify Item dialog box opens when you use either method.

Modify an item




2. Go to Step 4.

### *Step 4. Enter Specifications in the Item's Dialog Box*

**Step 4. Enter specifications in the item's dialog box:**

1. Select the General Information tab.

   **Note:** The Region ID field displays the active region to which you are adding, inserting or modifying this item and is read-only.

2. Enter the item location in the Location field.

   **Note:** A value of 0 will place the new item *after* the selected item; a value of –1 will place the item *before* the selected item.

3. In the Item ID field, type the item ID.
4. Select an Item Class ID from the drop-down list of configured item classes.
5. In the Reference ID field, type the reference ID for this item.
6. Select an Item Type from the drop-down list of configured item types.
7. Check the Int Hold box if you want to place an internal hold on the item.
8. Tab to the Reason field and type in an explanation for the hold.
9. Check the Ext Hold box if there is an external hold placed on the item by an external process in the region.

   **Note:** This will only affect the item indirectly.

10. Tab to the Reason field and type in an explanation for the hold.
11. Click **Apply** to effect changes.

## *Step 5. Set Attributes for an Item*

Associate attributes with the item using fields on the Attributes tab of the dialog box. You can select attributes and enter a value for each.

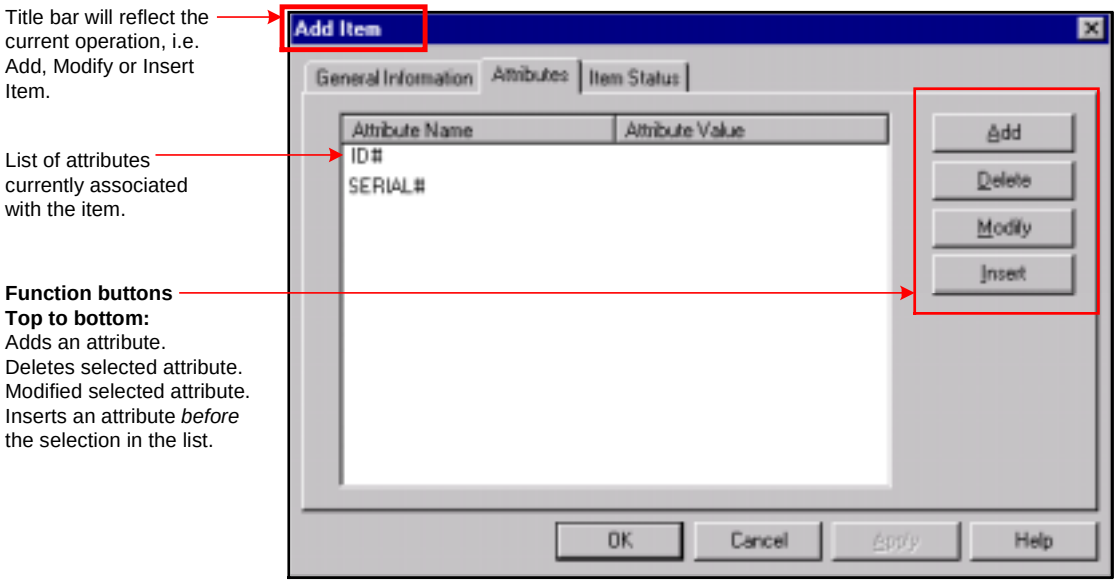
### Step 5. Set attributes for an item:

1. Select the Attributes tab to display the list of attributes that are currently associated with this item.

   **Note:** The window may not have any items listed.

2. Use the function buttons on the right of the dialog box to perform the following function. *See below for procedures.*
   - Add an attribute to the list.
   - Delete an attribute from the list.
   - Modify an attribute in the list.
   - Insert an attribute before an item in the list.

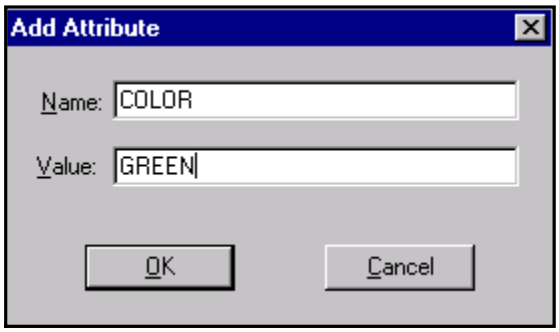
Set Attributes - Attributes tab



**Add an Attribute:**

1. Click **Add**

   The Add Attribute dialog box opens.

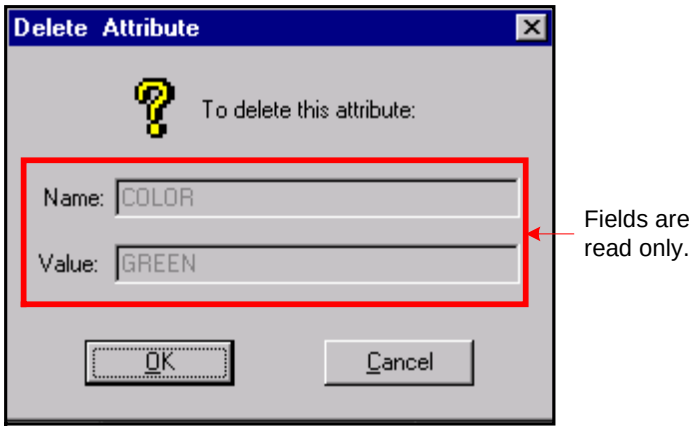



2. Enter the name of the attribute in the Name field.
3. Enter an associated value in the Value field.
4. Click **OK** to add the attribute to the list, or **Cancel** to cancel the operation.

**Delete an Attribute**

1. Select the attribute to be deleted from the list.
2. Click **Delete**.

   The Delete Attribute dialog box opens.

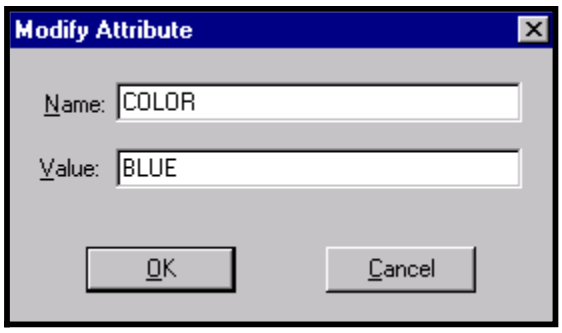



3. Click **OK** to delete the attribute, or click **Cancel** to cancel the operation.

### Modify an Attribute

1. Select the attribute to be modified from the list.
2. Click **Modify**.

   The Modify Attribute dialog box opens.



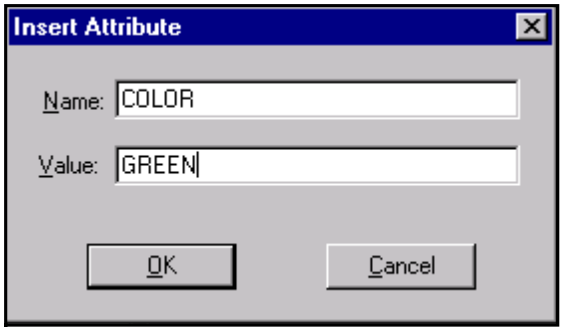

3. Do either or both:
   A. Change the Name
   B. Change the Value.
4. Click **OK** to modify this attribute, or click **Cancel** to cancel the operation.

### Insert an Attribute

1. Select an attribute on the list as the insertion point

   **Note:** Attributes are inserted before selected item.
2. Click **Insert**.

   The Insert Attribute dialog box opens.




3. Enter the name of the attribute to be inserted in the Name field.
4. Enter the associated value in the Value field.
5. Click **OK** to insert the attribute in the list, or click **Cancel** to cancel the operation.

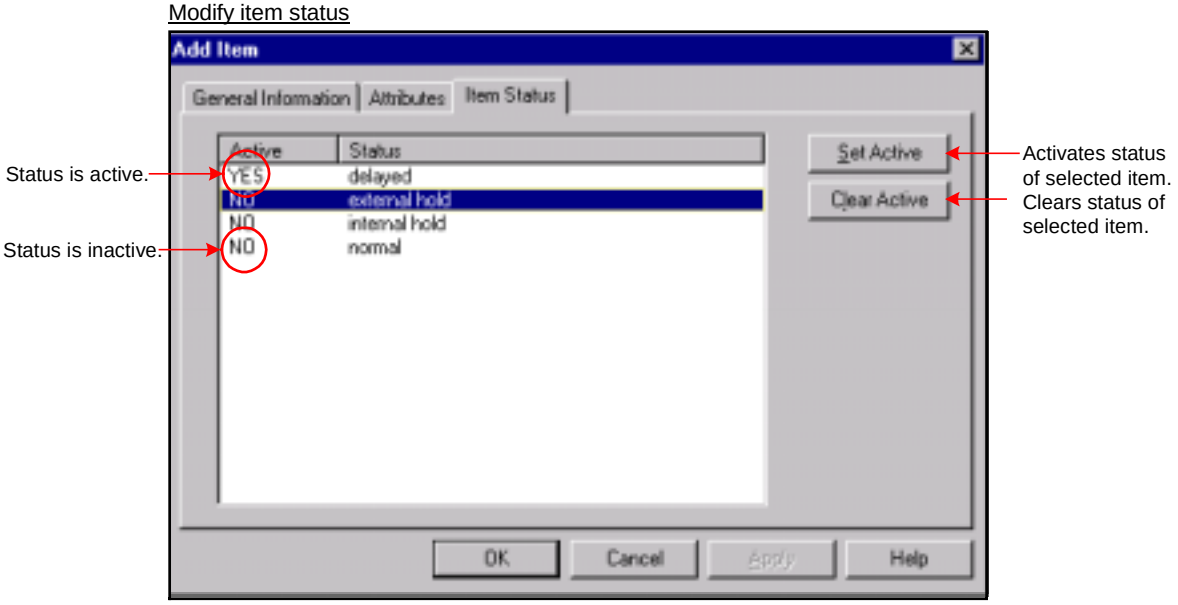
## *Step 6. View and Modify Status of Item*

**To view and modify the status of an item:**

1. Select the Item Status tab.

   The status list displays.

Modify item status




2. Select a status item in the list.
3. Do one of the following to set the status of the item:

| Click | To change the status to: |
|---|---|
| **Set Active** | Yes (activates item status). |
| **Clear Active** | No (deactivates item status). |

4. Click **OK** to effect Item Status change(s) or click **Cancel** to cancel the operation.

**Note:** The status list is configurable. *See "Configuring your Tracking Model" in the "Tracker Getting Started Guide," GFK-1694 for complete details.*

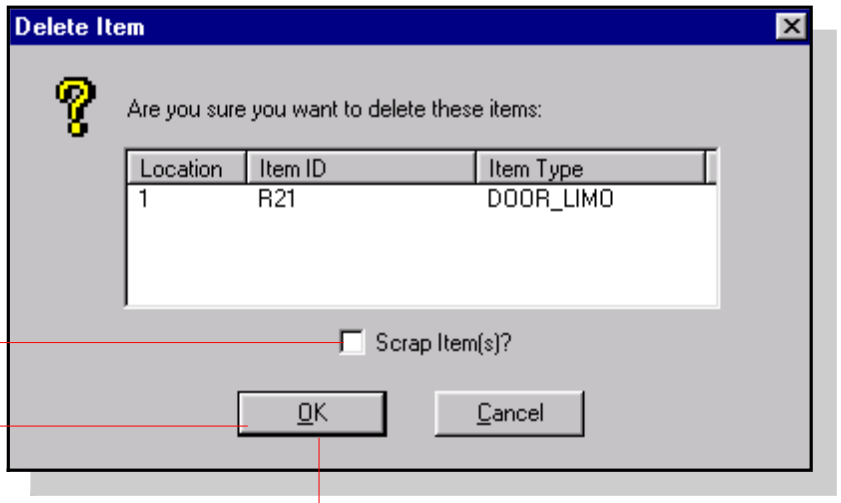
## Deleting an Item

**To delete an item from a region:**

1. Select the item in the list to be deleted.
2. Click the **Delete** button .

   The Delete Item dialog box opens.

Do one or the other...




To scrap the item, check option and click **OK**.

*—or—*

To delete the item, do not check Scrap option; click **OK**.

3. Do one of the following:

   *Option 1–Delete the item*

   A. Clear the Scrap Item(s) checkbox.

   B. Click **OK**.

   The item is deleted.

   *Option 2–Scrap the item*

   A. Check the Scrap Item(s) box.

   B. Click **OK**.

   The item is scrapped.

   *Option 3–Cancel the operation*

   Click **Cancel**.

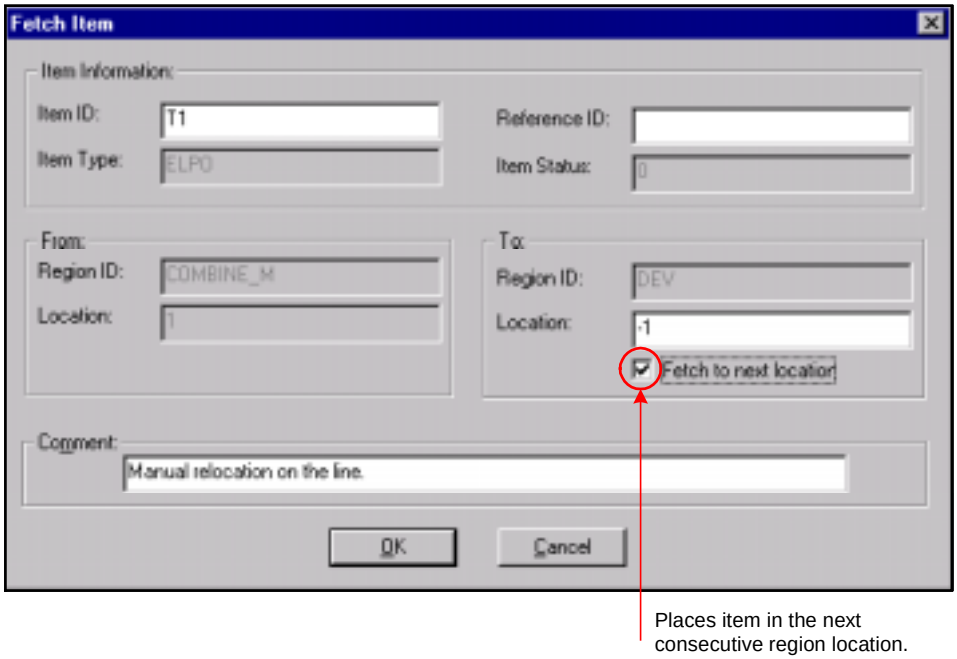
## Fetching an Item

To retrieve an item from one region and move it into the active region, use the Fetch Item function.

**To retrieve an item and move it into the active region:**

1. Click File on the menu bar.
2. Select Fetch.

   The Fetch Item dialog box opens.




3. Enter the ID of the item that you want to retrieve in the Item ID field, .
4. Enter the reference of the item you want to retrieve in the Reference ID field.
5. Indicate the region location where the item is to be placed in the Location field.
6. (*Optional*) Check Fetch to next location to place the item in the next consecutive location.

   **Note:** If you selected an item in the PRT window, the location will be automatically filled in.

**Example**

Active Region has the following occupied locations: L1, L2 and L3. Each location is capable of holding 2 items. Originally the region looks like the following:

| Location | Item |
|---|---|
| L1 | Item1 |
| L2 | Item2 |
| L3 | Item3 |

Retrieving a new item (Item4) into L3 in the region with the Fetch to next location box *unchecked* would result in the following:

| Location | Item |
|---|---|
| L1 | Item1 |
| L2 | Item2 |
| L3 | Item3 |
| L3 | Item4 (*Item is retrieved into the location to reside with the other item) |

Alternately, retrieving a new item (Item4) into the region with the Fetch to next location box *checked*, would result in the following:

| Location | Item |
|---|---|
| L1 | Item1 |
| L2 | Item2 |
| L3 | Item3 |
| L4 | Item4 (*Item is retrieved into the next consecutive location) |

6. (*Optional*) Enter any additional notes for the PRT Log file in the Comments field.
7. Click **OK** to move the item into the active region, or click **Cancel** to cancel the operation.

**Note:** PRT will locate the item in its current region; you are not required to know the Region ID.

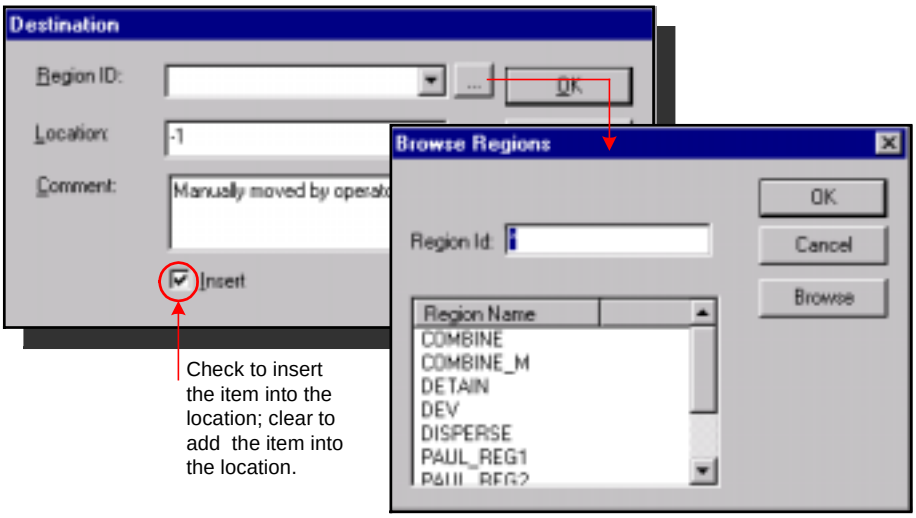
## Moving an Item

To move an item from one region to another region within the same project, or across projects, use the Move Item function.

### To move an item from the active region to another region:

1. Select the item to be moved.
2. Click **Move** to open the Destination dialog box.

Move an item




3. Select the destination region from the Region ID drop-down list, or type in the region ID, or click the **Browse** button to search for a region. *See "Browse Regions" topic at the beginning of this chapter for information on searching and valid criteria.*
4. In the Location field, type in where you want the item placed in the destination region.
5. (*Optional*) Type any additional notes for the PRT Log file in the Comment box.
6. Check the Insert box to insert the item into the location; clear to add the item to the location.

**Example:**

Destination Region has three (3) locations known as L1, L2 and L3. Each location is capable of holding 2 items. Originally the region looks like the following:

| Location | Item |
|---|---|
| L1 | Item1 |
| L2 | Item2 |
| L3 | Item3 |

Moving a new item (Item4) into L2 in the region with the Insert box **checked** would result in the following:

| Location | Item |
|---|---|
| L1 | Item1 |
| **L2** | **Item4** (*Item is inserted into the location in place of item) |
| L3 | Item3 |

Alternately, moving a new item (Item4) into the region with the Insert box **cleared**, would result in the following:

| Location | Item |
|---|---|
| L1 | Item1 |
| **L2** | **Item2** |
| **L2** | **Item4** (*Item is added to the location) |
| L3 | Item3 |

7. Click **OK** to move the item to the destination region, or click **Cancel** to cancel the operation.

## Advancing an Item

To advance an item or multiple items from the active region's exit location to the entrance location for a region along a specified route, use the Advance function.

**To advance an item(s) from the active region to another region:**

*Method 1 – Quick keys*

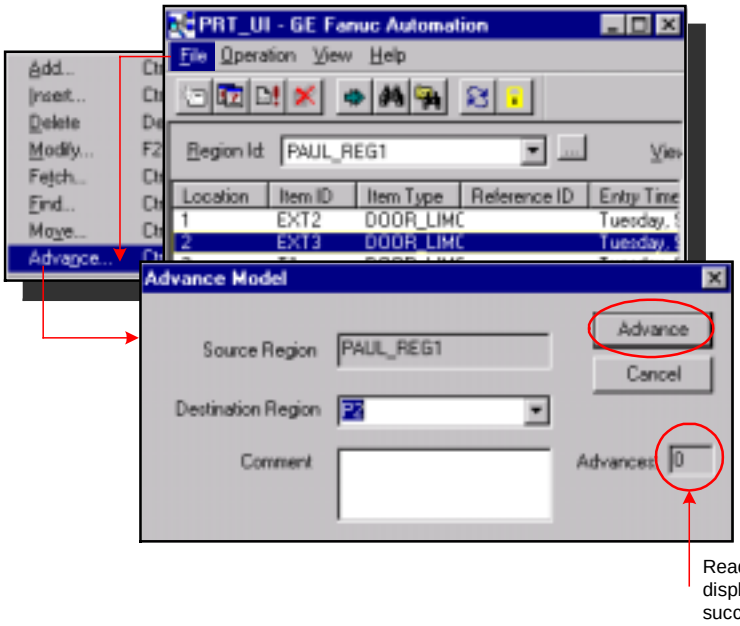
1. Select the item or items to be advanced.
2. Press **Ctrl+V**.

*Method 2 – Menu option*

1. Select the item or items to be advanced.
2. Click File on the menu bar.
3. Select Advance.

   The Advance Model dialog box opens.

Advance an item




**Tip:** Use the **Ctrl** key to select multiple items.

Continue from Methods 1 & 2

1. Select the Destination Region from the drop-down list, or type in the region ID.
2. (*Optional*) Type any additional notes for the PRT Log file in the Comment box.
3. Click **Advance** to advance the item(s) to the entrance location of the destination region. The number of items successfully advanced will display in the Advances field.
4. Click **Cancel** to close the dialog box.

**Note:** If there are no items in the Source region, or no routes configured for the Source region, a Warning message will display.

## Finding an Item

You can locate an item when you know the item or reference ID using the Find Item function.

**To find an item:**

1. Click **Find.**

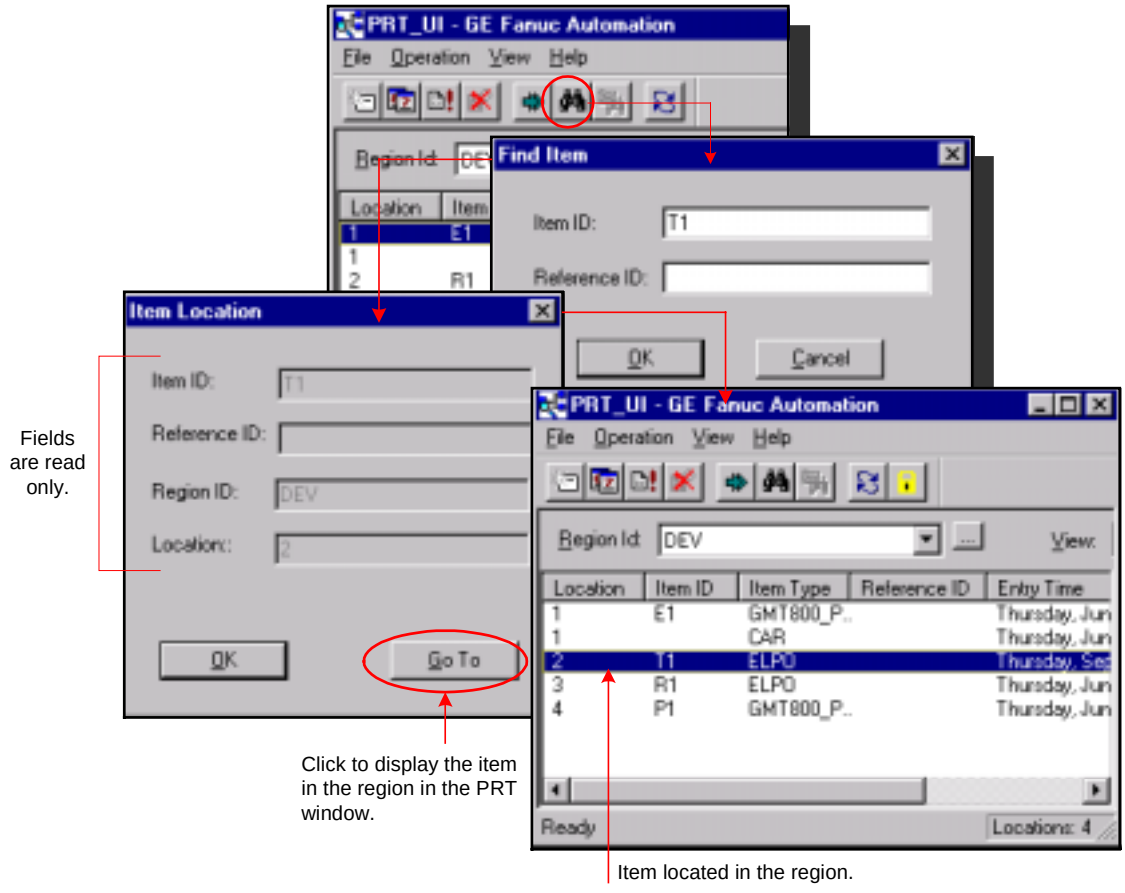
   The Find Item dialog box opens.

2. Provide information in one or both of the following fields:

| | |
|---|---|
| Item ID | ID of the item you want to find. |
| Reference ID | Reference ID of the item you want to find. |

3. Click **OK** to locate the item, or click **Cancel** to cancel the operation.

   **Note:** If the item cannot be located, a message box displays indicating the error. If the item is found, the Item Location box opens showing read-only information about the item.



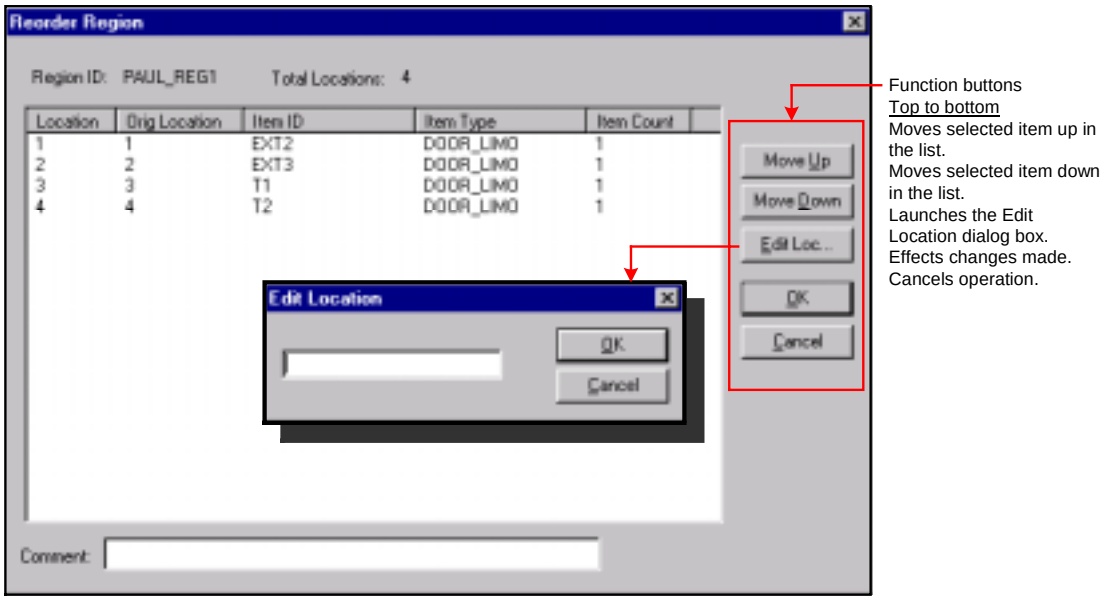

4. Click **OK** to exit the window, or **Go To** to display the item in the region in the PRT window.

## Reordering Items

You can reorder items in the active region using the Reorder Region function.

### To reorder items in a region:

1. Click Operation on the menu bar.
2. Select Reorder to open the Reorder Region dialog box.




3. Select the item to be moved in the list and proceed as follows:
   - Click **Move Up** to move the item up in the list or
   - Click **Move Down** to move the item down in the list.
4. Select the item in the list to move an item to a specific location or over a greater distance.
5. Click **Edit Loc** to open the Edit Location box.
6. Enter the location number where the item is to be moved in the field.

   **Example**

   In the Edit Location box, enter 18 to move the selected item to location number 18.
7. Click **OK**.
8. In the Comment field, type any additional notes to be saved to the PRT Log file.
8. Click **OK** to return to the PRT window and view changes, or click **Cancel** to exit without making any changes.

## Updating a Region's Status

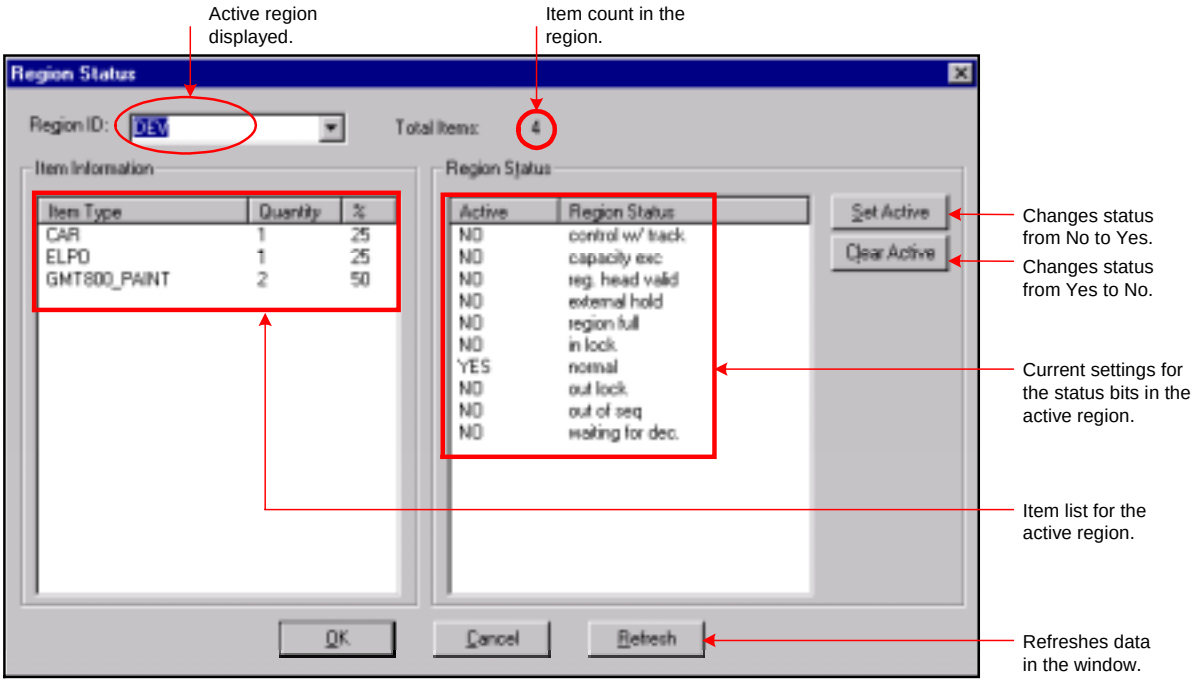
You can set the status bits for an entire region using the Region Status option. Region status fields are configured in the Tracker Configuration User Interface>PRT>Advanced>Status Codes.

*See the "Tracker Getting Started Guide,"GFK-1694 for information on status bits.*

### To set the status bits for a region:

1. Click Operation on the menu bar.
2. Select the Region Status.

   The Region Status dialog box opens.




3. To change the settings for a region, select the status bit from the list.
4. Click the appropriate button to set the status as needed:

| Button | To: |
|---|---|
| **Set Active** | Activate status by changing from No to Yes. |
| **Clear Active** | Deactivates status by changing from Yes to No. |

5. Click **OK** to effect changes to the region status, or click **Cancel** to cancel the operation.

# Auto Locking Regions

Regions that are configured to be lockable in the PRT Region Configuration dialog box must have certain status bits set when performing operations on items. The Auto Lock feature will automatically lock and retain the status bit settings for you whenever you perform operations on items within the region. You must turn on the auto lock feature to enable its functionality.

*See the "Tracker Getting Started Guide,"GFK-1694 for further details on the Auto Lock feature.*

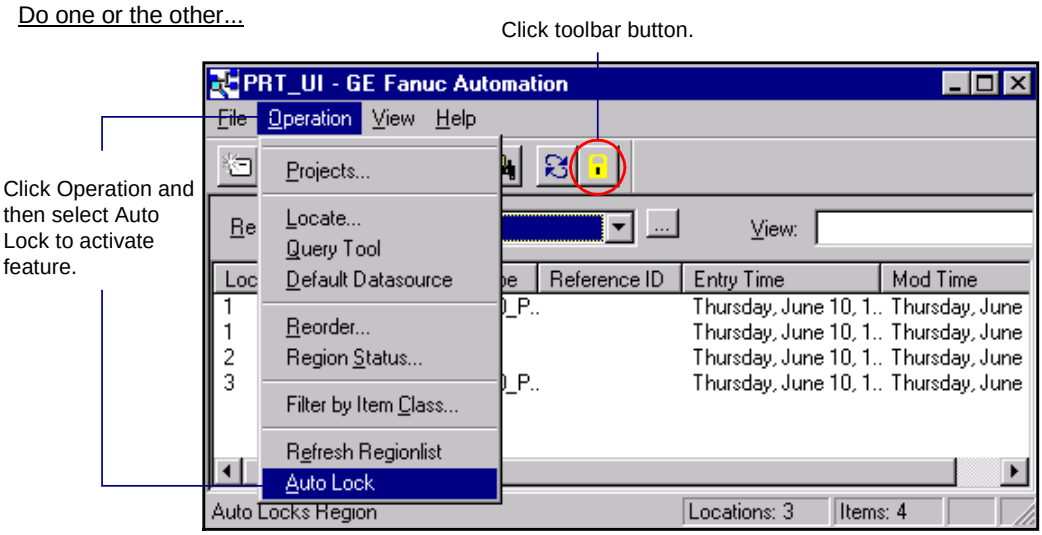
### To turn on the auto lock feature:

*Method 1 – Menu option*

1. Click Operation on the menu bar.
2. Select Auto Lock.

*Method 2 – Quickest*

Click .




***Result: Using either method a check mark displays next to Auto Lock on the Operation drop-down menu. The check mark indicates the feature is turned on.***

## *Operations and Setting Status Bits*

The operations affected by locked regions, and the status bit settings are described below:

| Operation | Source Region Status Bit | Destination Region Status Bit |
|---|---|---|
| Add Item | in_lock | N/A |
| Insert Item | in_lock | N/A |
| Delete Item | out_lock | in_lock |
| Fetch Item | out_lock | in_lock |
| Move Item | out_lock | in_lock |
| Advance Item | out_lock | in_lock |

# Printing Reports

Based on the current view in the PRT window, you can print a report with the same columns of information. The View drop-down list contains the different views configured you configured using the **prt_window.cfg** file. You can select the view that has the information you want to print, and then print out a report for your records. Use the print options to set up, preview and print out your report.

You can do the following:

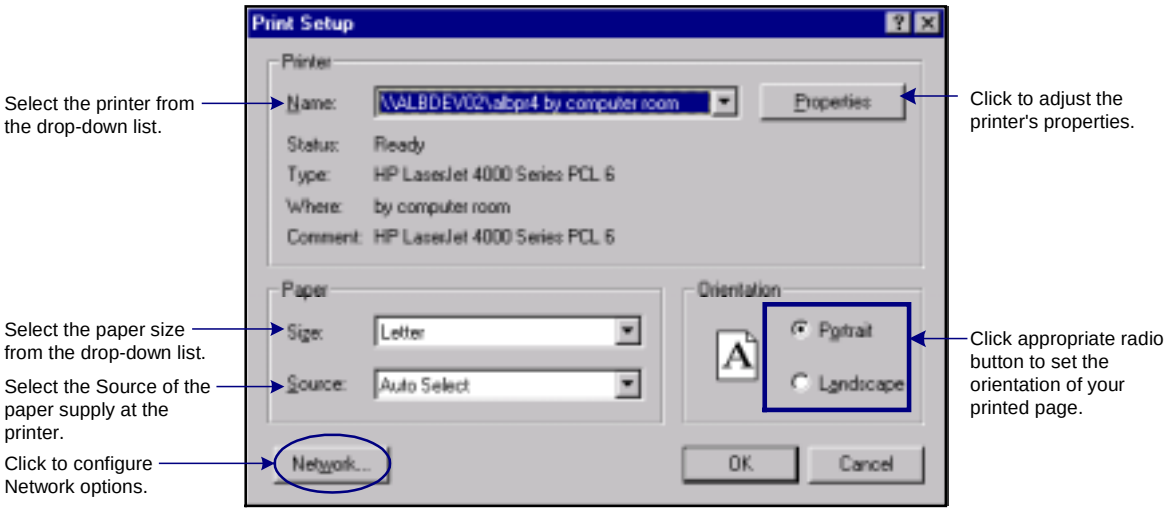
- Set up a printed report.
- Preview the printed report.
- Print the report.

## *Procedure to Set up a Printed Report*

**To set up printed report:**

1. Open the PRT window,
2. Select the view that displays the information you want to print from the View drop-down list.
2. Click File on the menu bar.
3. Select Print Setup.

   The Print Setup dialog box opens.




4. Set up the printer and page properties as follows:
   - Click **Properties** to set the printer properties.
   - Select the Size of the paper from the drop-down list.
   - Select the Source of the paper supply from the drop-down list.
   - Click **Network** to configure printer network options.
   - Check either the Portrait or Landscape radio button to set the orientation of the printed page.
5. Click **OK** to effect changes, or click **Cancel** to cancel the operation.

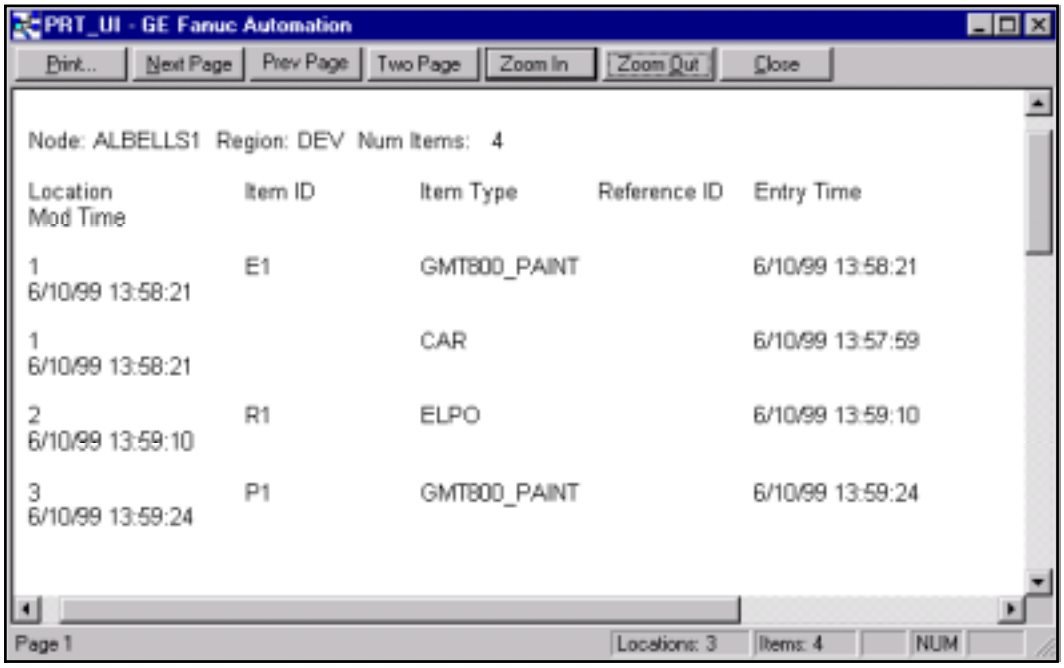
## *Procedure to Preview a Printed Report*

**To preview a printed report:**

1. Click File on the menu bar.
2. Select the Print Preview option to see how the printed report will look.

Function buttons Left to Right

- Prints report as you see it.
- Scrolls to the next page in the report.
- Scrolls to the previous page in the report.
- Previews two pages of the report.
- Enlarges the report on screen by percentages.
- Reduces the report on screen by percentages.
- Closes the print preview window.




3. The button bar at the top of the window performs the following functions:

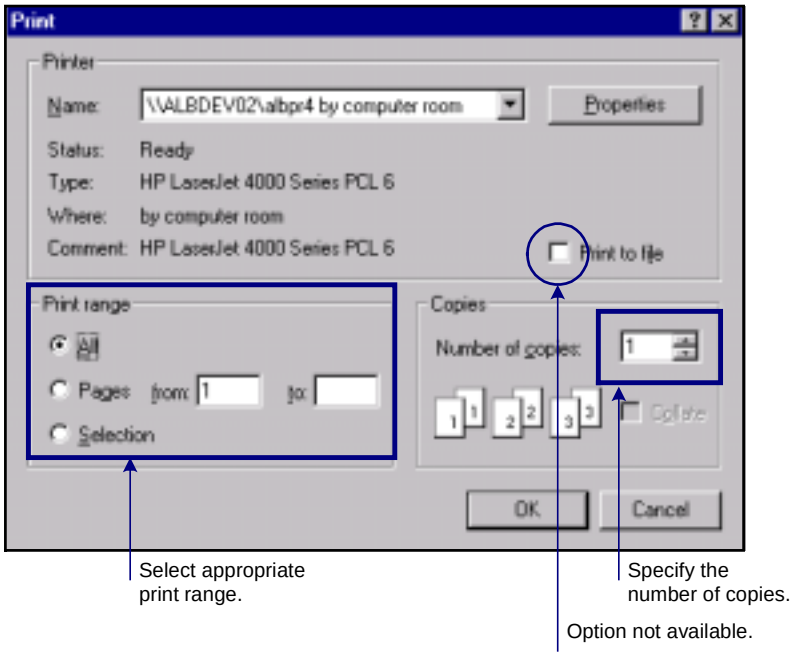
| Click | To |
|---|---|
| **Print** | Print out the report as it is displayed in view. |
| **Next Page** | Go to the next page of the report. |
| **Prev Page** | Go to the previous page in the report. |
| **Two Page** | View two pages in the viewing area. |
| **Zoom In** | Enlarge the view on screen. |
| **Zoom Out** | Reduce the view on screen. |
| **Close** | Exit print preview and return to the PRT window. |

**Tip:** If there is too much or not enough information in the report, use the View drop-down list to select another file configuration. If the information is not displayed properly or the way you want it, go to Print Setup to adjust the settings.

## *Procedure to Print a Report*

**To print a report:**

1. Click File on the menu bar.
2. Select Print option to open the Print dialog box.




3. Use the default printer or select another printer from the Name drop-down list.
4. Indicate the Print range as follows:
   A. Click All to print the entire report
   B. Click Pages and type the page range in the from and to boxes.
   C. Click Selection to print data previously selected in the PRT window. Note that this button is disabled when no data has been selected.
5. Indicate the Number of copies to print by typing in the number or by using the arrows to increase or decrease the number.
6. Click **OK** to print the indicated print range and number of copies, or click **Cancel** to cancel the operation.

# Configuration File Guide

## About the Configuration Files

This section is devoted to providing our customers with information and detailed instructions for using the Configuration Files. Before the implementation of the Tracker Configuration User Interface, the Tracking Model was configured using ASCII-text files. The need for this has been eliminated; however, for the convenience of our customers who are accustomed to editing these files, all of the information you need is included herein.

Handling configuration files includes the following:

- Identify the configuration files.
- Use the IDTPOP utility.
- Understand configuration file format and restrictions.
- Edit Configuration files based on provided definitions.

*See the "CIMPLICITY Tracker Getting Started Guide" for complete information about configuring Tracker using the Tracker Configuration User Interface*

### *Configuration File Acronyms*

For your reference, here is a list of acronyms that are used in this chapter.

| Acronym | Meaning |
|---|---|
| ISAM | Indexed Sequential Access Method |
| SCPOP | System Configuration Population |
| ASCII | American Standard Code for Information Interchange |

# Identify the Configuration Files

There are several required ISAM files to configure to run your tracking model, as well as several optional files for additional functionality. Tracker primarily uses binary coded files to run PRT. You can readily identify the files you want to work with by their **prt** prefix and .**dat** or **.cfg** extension.

### Guidelines for Required Configuration Files

The required configuration files ***must*** be configured in order to run your tracking model. Each file is described in detail below.

| File Name | File Definition | Description |
|---|---|---|
| `prt_sysdef` | PRT System Definitions | Sets the queue sizes for the ports used by the Data Collector, Data Service and Application Interface subsystems. Minimum recommended size is 10 for each port. |
| `prt_service` | PRT Service Record | Identifies each PRT Service in the tracking model. |
| `prt_intproc` | PRT Interested Process/Data Provider Mapping | Enables related processes to receive updates from the tracking model. Both prt_grd and prt_stnd require this file to be configured in order to function properly. |
| `prt_group` | PRT Group | Identifies the Groups in your tracking model. Because every region must belong to a group, this file must be configured. |
| `prt_region` | PRT Region | Identifies the regions in your tracking model, their primary and secondary types and other important information. |
| `prt_grp_reg` | PRT Group Tracking Region | Ties the tracking regions to their tracking groups. |
| `prt_item_typ` | PRT Item Type | Defines the item types, classes and parent-child relationships between items. |
| `prt_route` | PRT Route | Identifies routes between regions and associations with transfer points. |
| prt_`attributes`.cfg | PRT Attribute Labels | Configures a set of attributes for all items or for items in a specific class. This is not a required file, but is recommended. |

**Note:** All of the above files have a **.dat** extension, with the exception of the last one, which has a **.cfg** extension. Files with a .dat extension are binary; files with a .cfg extension are in text format.

### Guidelines for Optional Configuration Files

The optional configuration files are not required to run a basic tracking model. However the configuration of your tracking model may require some or all of these files in order to provide you with the kind of information you want to collect. Each file is described in detail below:

| File Name | File Definition | Description |
|---|---|---|
| `prt_type_trn` | PRT Item Type Translation | Translates item type codes received from detection devices into item type IDs that can be recognized by Tracker. This file must be configured when the prt_route records have a translation type value of 2 |
| `prt_inv_type` | PRT Route Invalid Type | Identifies an item type(s) that is invalid for a specified route. |
| `prt_assoc` | PRT Region Item Association | Defines the item type roles for parent/child associations, of which there is one record configured for each item type. |
| `prt_status` | PRT Status Code | Determines the status bits in the track model for both regions and items. |
| `prt_disp_pt` | PRT Region Item Type Display Point | Configures the item type quantity in a region that is to be displayed through a setpoint.<br>Note: Must be configured when prt_grd is configured. |
| `prt_att_pt` | PRT Region Item Attribute Display Point | Identifies the setpoint ID that will link item attributes by region.<br>Note: Must be configured when prt_grd is configured. |
| `prt_reg_at` | PRT Region Attribute Counter Definitions | Indicates the counters in item attributes. This will in turn trigger the action taken when an item containing the defined attribute enters the region. |
| `prt_stnd.cfg` | PRT Standby Process | Determines the PRT data collector that the standby process will monitor. |
| `prt_window.cfg` | PRT Field Labels | Defines the column names for the PRT_UI (PRT User Interface) in place of attribute names. A default record is provided with Tracker and must be present to run the PRT_UI. |

**Note:** All of the above files have a **`.dat`** extension, with the exception of the last two, which have a **`.cfg`** extension. Files with a **`.dat`** extension are binary; files with a **`.cfg`** extension are in text format.

# Use the IDTPOP Utility

In order to edit the binary-coded configuration files, they must be converted to a format that can be edited. The IDTPOP Utility enables you to convert the binary code to ASCII format and edit using the text editor of your choice.

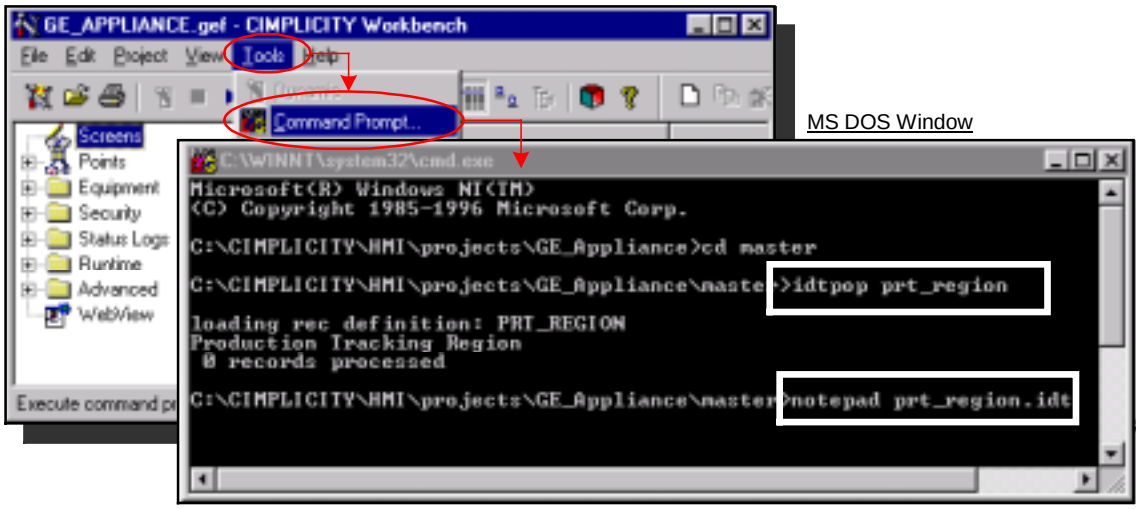
### To convert files to ASCII format using IDTPOP:

1. Click Tools on the CIMPLICITY Workbench menu bar.
2. Select Command Prompt to open a DOS window.
3. On the Command line, type **`cd master`**.
4. Press **ENTER** to switch to the Master directory.
5. Type **`idtpop`** <*filename*>, e.g. **`idtpop prt_region`**.
6. Press **ENTER**.

***Result: An indication of the number of records processed will display, and the file will be converted to ASCII text.***

Project Workbench

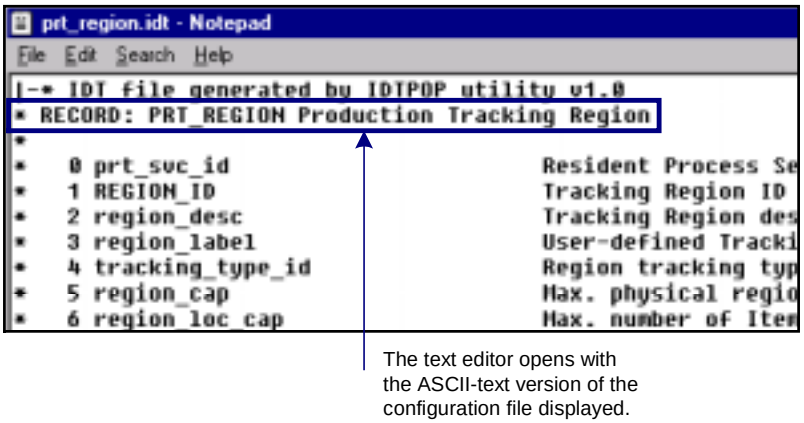
MS DOS Window




7. Type <**`text editor`**> <**`filename.idt`**>, e.g. **`notepad prt_region.idt`**.
8. Press **ENTER**.

***Result: The text editor will open with the ASCII-text version of the file displayed.***

IDT file displayed in text editor


prt_region.idt - Notepad
File Edit Search Help
|-* IDT file generated by IDTPOP utility v1.0
* RECORD: PRT_REGION Production Tracking Region
*
* 0 prt_svc_id Resident Process Se
* 1 REGION_ID Tracking Region ID
* 2 region_desc Tracking Region des
* 3 region_label User-defined Tracki
* 4 tracking_type_id Region tracking typ
* 5 region_cap Max. physical regio
* 6 region_loc_cap Max. number of Ite
The text editor opens with the ASCII-text version of the configuration file displayed.

# About Configuration File Setup

Each configuration file has a set of reserved characters and restrictions that you must observe. Once you understand how the files are set up, you will have no trouble configuring your tracking model.

Details about configuration file setup include:

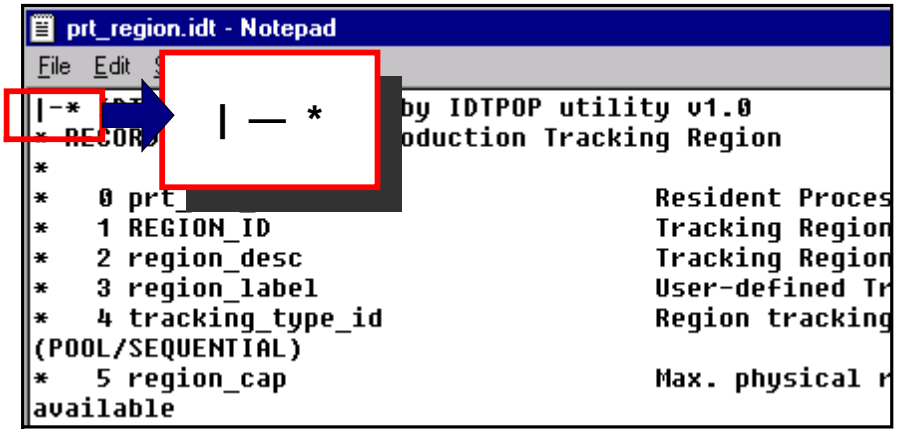
- Reserved characters
- Restrictions
- Configuration file format

## *Configuration File Reserved Characters*

The first line in the **`.idt`** file must define three reserved characters. All other characters in the line are ignored. You may select characters other than the ones used here; however, avoid using characters that will be used in the data fields.

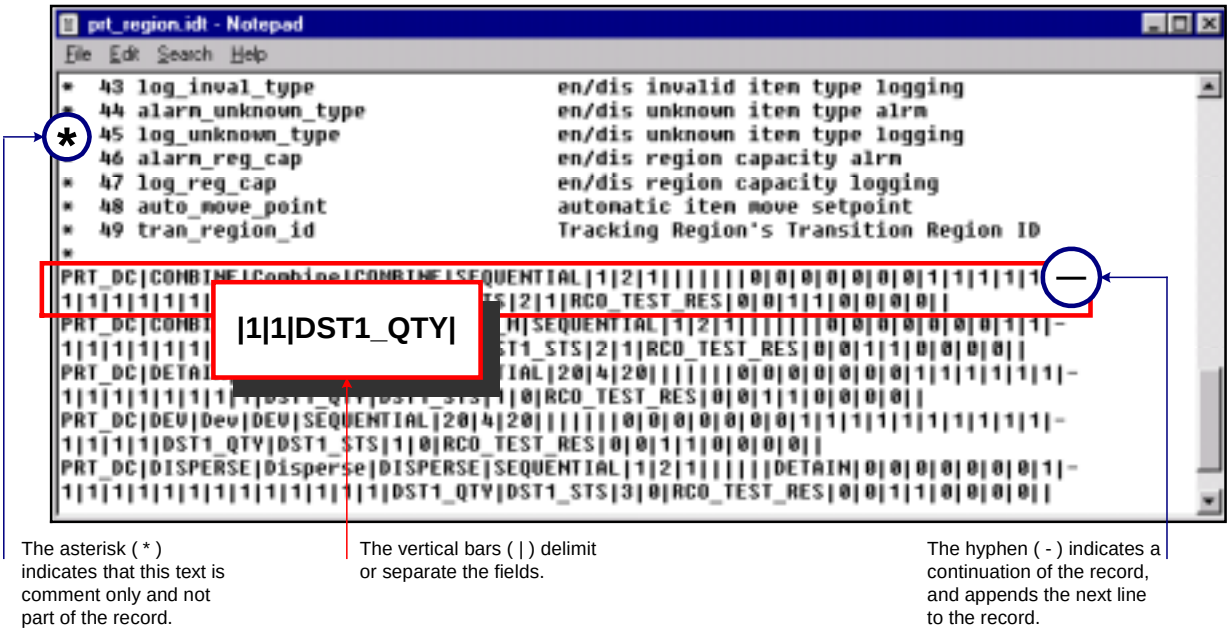
- The first character defines the **delimiter**. This character will be used to separate fields in the record. In the example below, the vertical bar ( | ) is used for a delimiter.
- The second character is used to show a **continuation** of a record. When a record exceeds the length of one line, this character is used to append the next line to the record. In the example below, the hyphen ( - ) is used for a continuation character.
- The third character identifies the **comment** character. When the comment character is used before text, it is not recognized as part of the record, but rather additional information for the user. In the example below, the asterisk ( * ) is used for a comment character.

Reserved characters: delimiter, continuation and comment.



**Example of Delimiter, Continuation and Comment Characters:**

IDT file in text editor




The asterisk ( * ) indicates that this text is comment only and not part of the record.

The vertical bars ( | ) delimit or separate the fields.

The hyphen ( - ) indicates a continuation of the record, and appends the next line to the record.

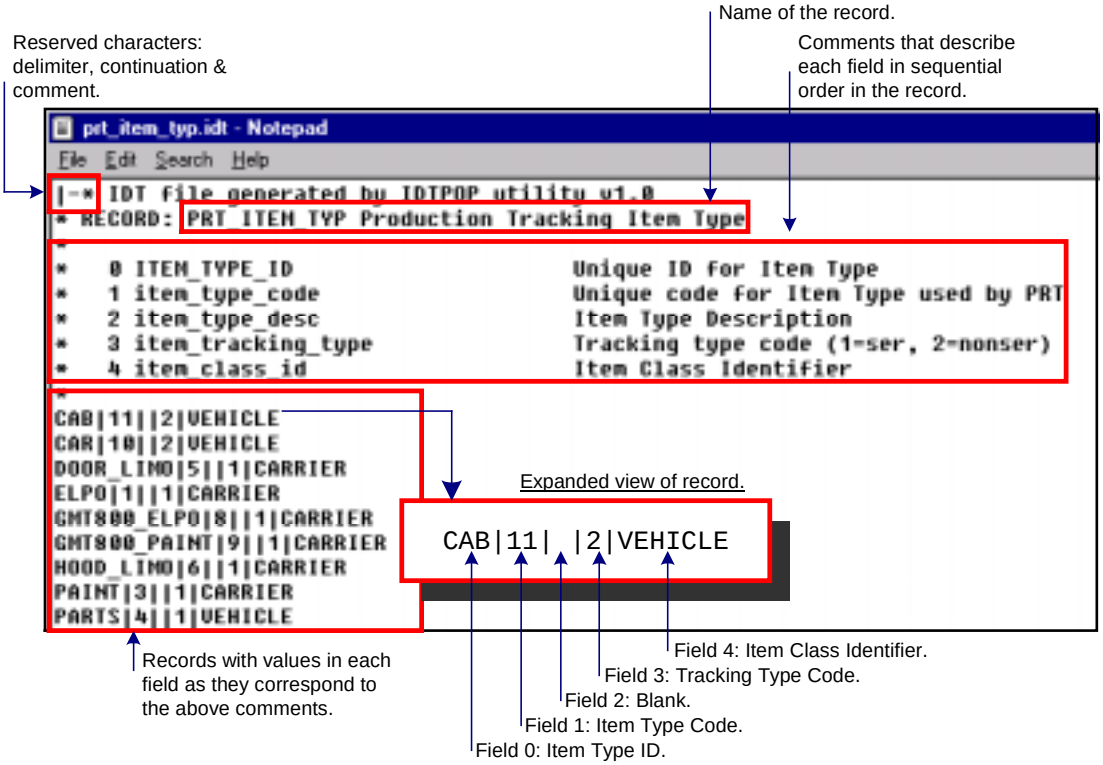
## *Configuration File Restrictions*

The following restrictions must be observed when editing the configuration files.

| | |
|---|---|
| Delimiter | All fields must be separated by a delimiter, which is the first character defined in the first line of the file. |
| Field length | Must be less than or equal to its configured maximum.<br>Cannot exceed 250 characters. |
| Field Order | The fields must be in the same order in the record as they appear in the input file. |
| Field Type | Vary depending on the file being edited; enter values appropriate for the file. |
| Field Values | Cannot exceed the specified size of the field.<br>Do not support embedded spaces; use underscore (_) character instead.<br>**Exception:** Text or description fields support embedded spaces. |
| Fields | Empty fields are supported, but must be delimited in the record. |
| Logical line length | Must be less than or equal to 2000 characters.<br>Includes all continued lines in the record. |
| Padding | Character fields are blank-padded automatically when the value is less than the field size.<br>Numeric fields are zero-filled automatically when the value is less than the field size. |
| Physical line length | Must be less than or equal to 250 characters. |

### *Configuration File Format*

When the ASCII-text version of the file is viewed in a text editor, you will notice that they all have the same basic format. The example below provides a detailed look at the format of a configuration file.


Reserved characters: delimiter, continuation & comment.
Name of the record.
Comments that describe each field in sequential order in the record.
prt_item_typ.idt - Notepad
File Edit Search Help
|-* IDT file generated by IDTPOP utility v1.0
* RECORD: PRT_ITEM_TYP Production Tracking Item Type
*
* 0 ITEM_TYPE_ID Unique ID for Item Type
* 1 item_type_code Unique code for Item Type used by PRT
* 2 item_type_desc Item Type Description
* 3 item_tracking_type Tracking type code (1=ser, 2=nonser)
* 4 item_class_id Item Class Identifier
*
CAB|11||2|VEHICLE
CAR|10||2|VEHICLE
DOOR_LIMO|5||1|CARRIER
ELPO|1||1|CARRIER
GMT800_ELPO|8||1|CARRIER
GMT800_PAINT|9||1|CARRIER
HOOD_LIMO|6||1|CARRIER
PAINT|3||1|CARRIER
PARTS|4||1|VEHICLE
Expanded view of record.
CAB|11| |2|VEHICLE
Records with values in each field as they correspond to the above comments.
Field 0: Item Type ID.
Field 1: Item Type Code.
Field 2: Blank.
Field 3: Tracking Type Code.
Field 4: Item Class Identifier.

# Configuration File Definitions

A definition is provided for each configuration file to include field name, description, data type and an entry example. Use the definitions to edit configuration files as needed for your Tracking Model.

The following configuration file definitions are covered:

- PRT_Sysdef
- PRT_Service
- PRT_Intproc
- PRT_Group
- PRT_Region
- PRT_Grp_Reg
- PRT_Item_Typ
- PRT_Route
- PRT_Type_Trn
- PRT_Inv_Type
- PRT_Assoc
- PRT_Status
- PRT_Disp_Pt
- PRT_Att_Pt
- PRT_Reg_At
- PRT_Attributes.cfg
- PRT_Stnd.cfg
- PRT_Window.cfg

# PRT_SYSDEF

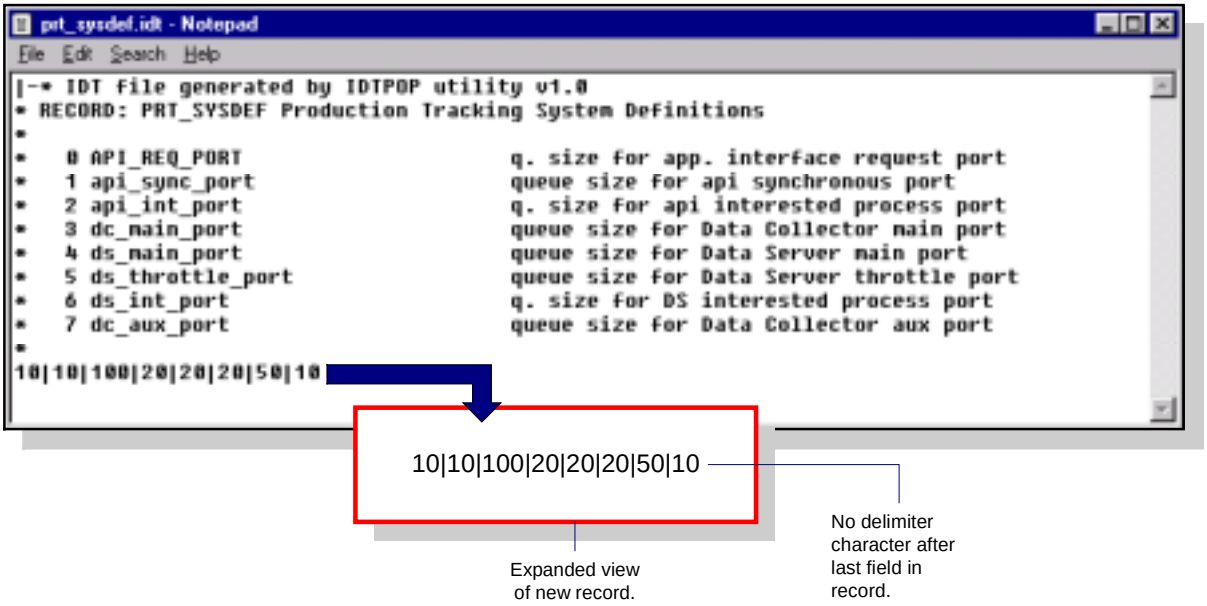
The PRT System Definitions (**prt_sysdef**) configuration file is used to define the queue sizes for the ports used by the Data Collector, Data Server and Application Interface subsystems. Each of the fields is described below, complete with entry examples and a sample configuration file.

**REQUIRE FILE**

| No. | Field Name | Description | Data Type | Entry Example |
|---|---|---|---|---|
| 0 | **api_req_port** | Sets the queue size for the Application Interface's request port. This is used to receive requests for item and item hold data from the PRT data collector(s). | Integer | **10** |
| 1 | **api_sync_port** | Sets the queue size for the Application Interface's synchronous port. This is used to send requests to PRT data collector(s), and to send requests and receive responses from the PRT data server(s). | Integer | 10 |
| 2 | **api_int_port** | Sets the queue size for the Application Interface's interested process port. This is used to receive interested process updates from the PRT data server(s). | Integer | 10 |
| 3 | **dc_main_port** | Sets the queue size for the data collector's main port. | Integer | 20 |
| 4 | **ds_main_port** | Sets the queue size for the data server's main port. | Integer | 20 |
| 5 | **ds_throttle_port** | Sets the queue size for the data server's throttle port. | Integer | 20 |
| 6 | **ds_int_port** | Sets the queue size for the data server's interested process port. | Integer | 50 |
| 7 | **dc_aux_port** | Sets the queue size for the data collector's auxiliary port. | Integer | 10 |

**Important:** GE Fanuc recommends that the ports in this record be set to a minimum value of 10.

### *Configuration File Example*


prt_sysdef.idt - Notepad
File Edit Search Help
|-* IDT File generated by IDTPOP utility v1.0
* RECORD: PRT_SYSDEF Production Tracking System Definitions
*
* 0 API_REQ_PORT q. size for app. interface request port
* 1 api_sync_port queue size for api synchronous port
* 2 api_int_port q. size for api interested process port
* 3 dc_main_port queue size for Data Collector main port
* 4 ds_main_port queue size for Data Server main port
* 5 ds_throttle_port queue size for Data Server throttle port
* 6 ds_int_port q. size for DS interested process port
* 7 dc_aux_port queue size for Data Collector aux port
*
10|10|100|20|20|20|50|10
10|10|100|20|20|20|50|10
Expanded view of new record.
No delimiter character after last field in record.

## PRT_SERVICE

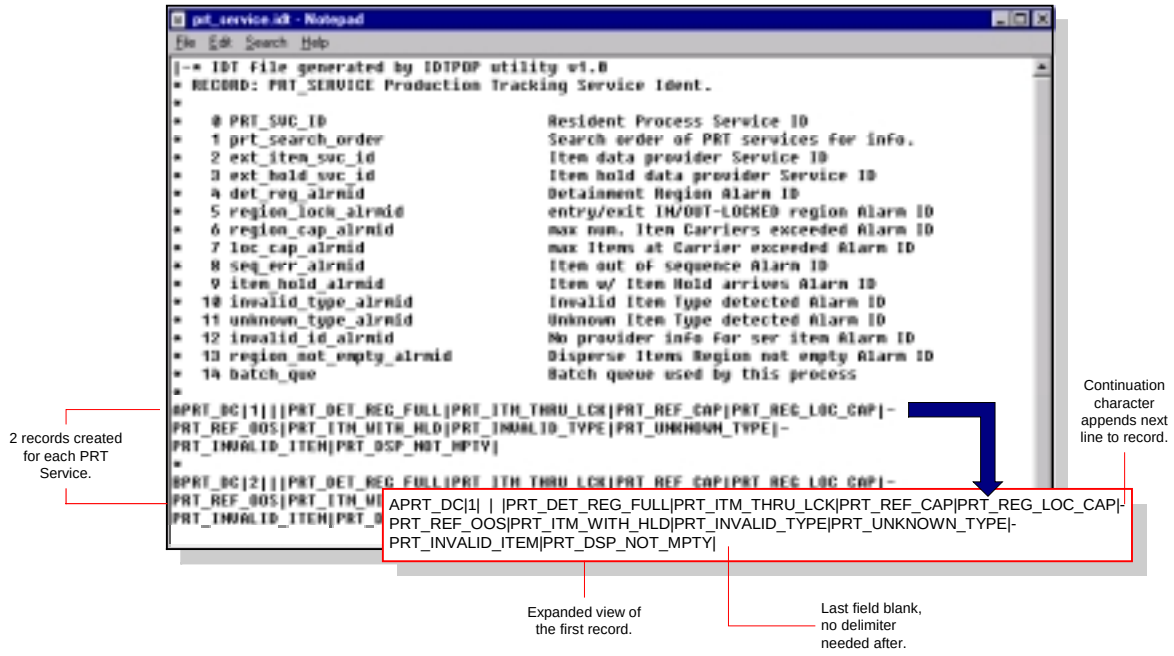
The PRT Service (**prt_service**) configuration file is used to identify each Production Tracking Service. Each of the fields is described below, complete with entry examples and a sample configuration file.

**REQUIRED FILE**

| No. | Field Name | Description | Data Type | Entry Example |
|---|---|---|---|---|
| 0 | **prt_svc_id** | Specifies the PRT Data Collector Resident Process Service ID. This must be identical to the ID in **service.dat**. | 32 characters | **APRT_DC** |
| 1 | **prt_search_order** | Indicates the order in which this service will be searched for information relative to other configured PRT Services. | 1 byte | 1 |
| 2 | **ext_item_svc_id** | Service ID of the external process providing item tracking data. | 32 characters | Blank |
| 3 | **ext_hold_svc_id** | Service ID of the external process providing hold information | 32 characters | Blank |
| 4 | **det_reg_alrmid** | Alarm ID for items placed in detainment region. | 32 characters | **PRT_DET_ REF_FULL** |
| 5 | **region_lock_alrmid** | Alarm ID generated when an item enters IN-LOCKED region, or exit OUT-LOCKED region via a point-based move. | 32 characters | **PRT_ITM_ THRU_LCK** |
| 6 | **region_cap_alrmid** | Alarm ID when the maximum number of item carriers in the region are exceeded. | 32 characters | **PRT_REG_ CAP** |
| 7 | **loc_cap_alrmid** | Alarm ID when the maximum number of items at a single region location / item carrier have been exceeded. | 32 characters | **PRT_REF_ LOC_CAP** |

| No. | Field Name | Description | Data Type | Entry Example |
|---|---|---|---|---|
| 8 | **seq_err_alrmid** | Alarm ID when an item is out of sequence at a region exit transition indicator. | 32 characters | **PRT_REG_ OOS** |
| 9 | **item_hold_alrmid** | Alarm ID when an item arrives in a tracking region and is determined to have an item-hold specification. | 32 characters | **PRT_ITM_ WITH_HLD** |
| 10 | **invalid_type_ alrmid** | Alarm ID when an invalid item type is detected. | 32 characters | **PRT_INVALID_ TYPE** |
| 11 | **unknown_type_ alrmid** | Alarm ID when an unknown item type is detected. | 32 characters | **PRT_ UNKNOWN_ TYPE** |
| 12 | **invalid_id_alrmid** | Alarm ID when a serialized item enters a region and information cannot be retrieved for that item. | 32 characters | **PRT_INVALID_ ITEM** |
| 13 | **region_not_empty_ alrmid** | Alarm ID when all items have not left a disperse tracking region and new items have arrived in the region. | 32 characters | **PRT_DSP_ NOT_MPTY** |
| 14 | **Batch_queue** | NOT USED | Not used | Blank |

### *Configuration File Example*

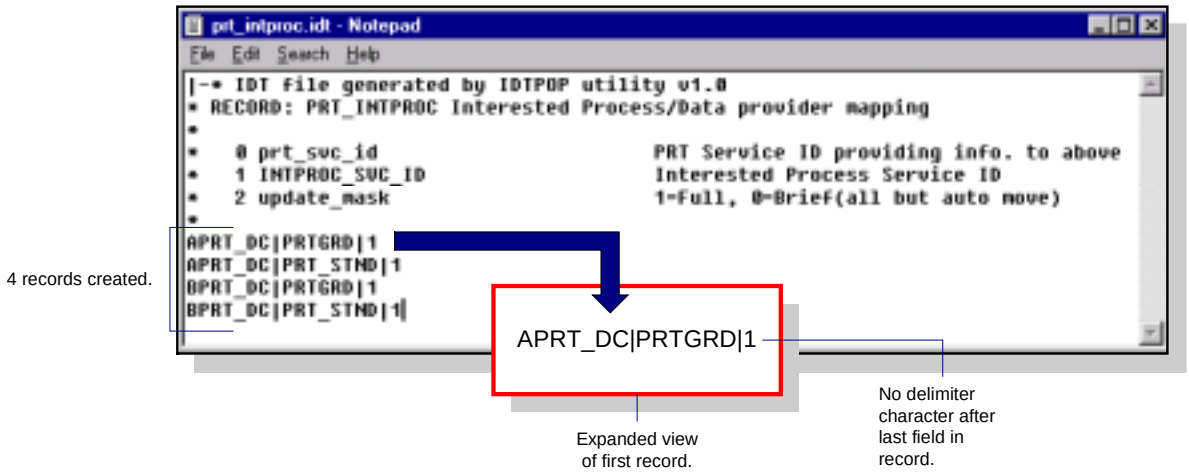

prt_service.idt - Notepad
File Edit Search Help
|-* IDT File generated by IDTPDP utility v1.0
* RECORD: PRT_SERVICE Production Tracking Service Ident.
*
* 0 PRT_SVC_ID Resident Process Service ID
* 1 prt_search_order Search order of PRT services for info.
* 2 ext_item_svc_id Item data provider Service ID
* 3 ext_hold_svc_id Item hold data provider Service ID
* 4 det_reg_alrmid Detainment Region Alarm ID
* 5 region_lock_alrmid entry/exit IN/OUT-LOCKED region Alarm ID
* 6 region_cap_alrmid max num. Item Carriers exceeded Alarm ID
* 7 loc_cap_alrmid max Items at Carrier exceeded Alarm ID
* 8 seq_err_alrmid Item out of sequence Alarm ID
* 9 item_hold_alrmid Item w/ Item Hold arrives Alarm ID
* 10 invalid_type_alrmid Invalid Item Type detected Alarm ID
* 11 unknown_type_alrmid Unknown Item Type detected Alarm ID
* 12 invalid_id_alrmid No provider info for ser item Alarm ID
* 13 region_not_empty_alrmid Disperse Items Region not empty Alarm ID
* 14 batch_que Batch queue used by this process
*
#PRT_DC|1|||PRT_DET_REG_FULL|PRT_ITM_THRU_LCK|PRT_REF_CAP|PRT_REG_LOC_CAP|-
PRT_REF_OOS|PRT_ITM_WITH_HLD|PRT_INVALID_TYPE|PRT_UNKNOWN_TYPE|-
PRT_INVALID_ITEM|PRT_DSP_NOT_MPTY|
*
#PRT_DC|2|||PRT_DET_REG_FULL|PRT_ITM_THRU_LCK|PRT_REF_CAP|PRT_REG_LOC_CAP|-
2 records created for each PRT Service.
Continuation character appends next line to record.
APRT_DC|1| | |PRT_DET_REG_FULL|PRT_ITM_THRU_LCK|PRT_REF_CAP|PRT_REG_LOC_CAP|-
PRT_REF_OOS|PRT_ITM_WITH_HLD|PRT_INVALID_TYPE|PRT_UNKNOWN_TYPE|-
PRT_INVALID_ITEM|PRT_DSP_NOT_MPTY|
Expanded view of the first record.
Last field blank, no delimiter needed after.

# PRT_INTPROC

The PRT Interested Process / Data Provider Mapping (**prt_intproc**) configuration file is used to map interested processes to the PRT Services. This will serve to supply the interested processes with regular updates. Each of the fields is described below, complete with entry examples and a sample configuration file.

| No. | Field Name | Description | Data Type | Entry Example |
|---|---|---|---|---|
| 0 | **prt_svc_id** | Unique identifier of a PRT service that will provide interested process updates. | 32 characters | **APRT_DC** |
| 1 | **intprc_svc_id** | Unique identifier of an interested process service ID. | 32 characters | **PRTGRD** |
| 2 | **update_mask** | Specifies the type of updates to be supplied to the interested process. Values are 1=Full; 0=Brief | Long word | **1** |

### *Configuration File Example*


prt_intproc.idt - Notepad
File Edit Search Help
|-* IDT file generated by IDTPDP utility v1.0
* RECORD: PRT_INTPROC Interested Process/Data provider mapping
*
*  0 prt_svc_id  PRT Service ID providing info. to above
*  1 INTPROC_SVC_ID  Interested Process Service ID
*  2 update_mask  1-Full, 0-Brief(all but auto move)
*
APRT_DC|PRTGRD|1
APRT_DC|PRT_STND|1
BPRT_DC|PRTGRD|1
BPRT_DC|PRT_STND|1
4 records created.
APRT_DC|PRTGRD|1
Expanded view of first record.
No delimiter character after last field in record.

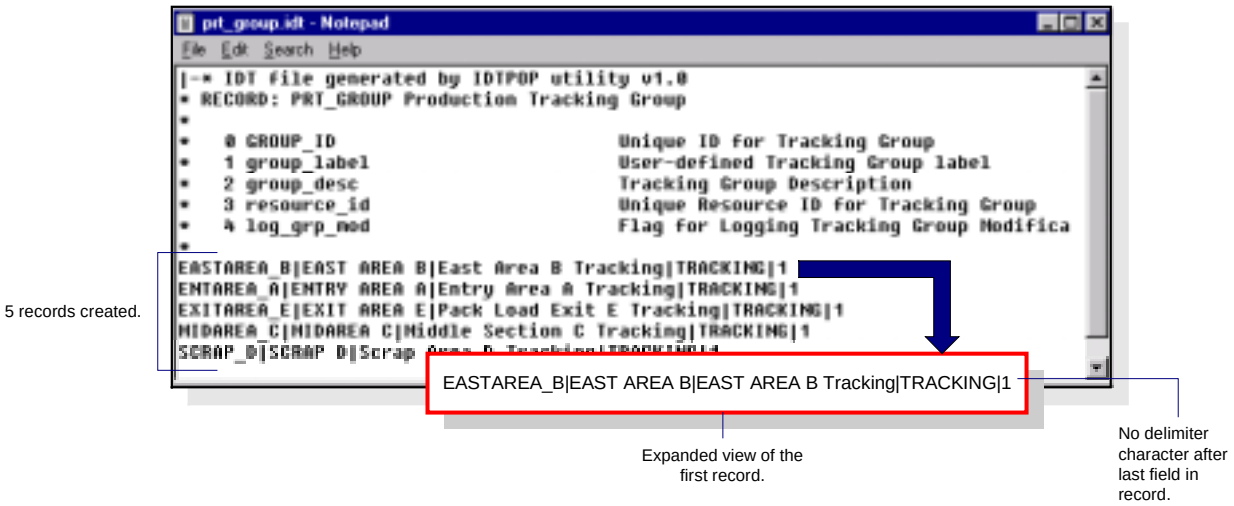
# PRT_GROUP

The PRT Group (**prt_group**) configuration file is used to define each of the Tracking Groups. Every tracking region must belong to a Group. Each of the fields is described below, complete with entry examples and a sample configuration file.

**REQUIRED FILE**

| No. | Field Name | Description | Data Type | Entry Example |
|---|---|---|---|---|
| 0 | **group_id** | Unique name for the tracking group. | 16 characters | EASTAREA_B |
| 1 | **group_label** | Text to appear with the tracking region ID for user recognition. | 16 characters | EAST AREA B |
| 2 | **Group_desc** | Description of the tracking group. | 40 characters | EAST AREA B Tracking |
| 3 | **resource id** | Unique name of the resource for the group. | 16 characters | TRACKING |
| 4 | **Log_grp_mod** | Flag for logging tracking group modifications, i.e. change of hold status on group. | One byte | 1 |

### *Configuration File Example*


prt_group.idt - Notepad
File Edit Search Help
|-* IDT file generated by IDTPOP utility v1.0
* RECORD: PRT_GROUP Production Tracking Group
*
* 0 GROUP_ID Unique ID for Tracking Group
* 1 group_label User-defined Tracking Group label
* 2 group_desc Tracking Group Description
* 3 resource_id Unique Resource ID for Tracking Group
* 4 log_grp_mod Flag for Logging Tracking Group Modifica
*
EASTAREA_B|EAST AREA B|East Area B Tracking|TRACKING|1
ENTAREA_A|ENTRY AREA A|Entry Area A Tracking|TRACKING|1
EXITAREA_E|EXIT AREA E|Pack Load Exit E Tracking|TRACKING|1
MIDAREA_C|MIDAREA C|Middle Section C Tracking|TRACKING|1
SCRAP_D|SCRAP D|Scrap
5 records created.
EASTAREA_B|EAST AREA B|EAST AREA B Tracking|TRACKING|1
Expanded view of the first record.
No delimiter character after last field in record.

# PRT_REGION

The PRT Region (**prt_region**) configuration file is used to define every region in the Tracking Model. Each of the fields is described below, complete with entry examples and a sample configuration file.

**REQUIRED FILE**

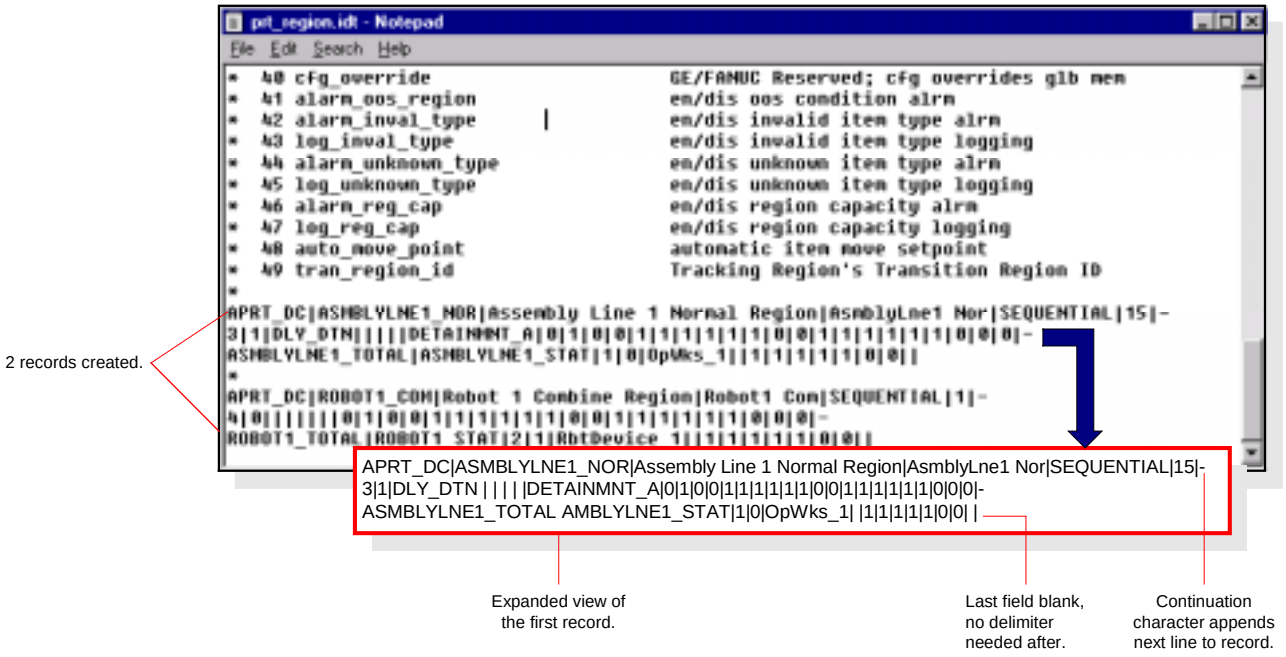
| No. | Field Name | Description | Data Type | Entry Example |
|---|---|---|---|---|
| 0 | **prt_svc_id** | Specifies the PRT Data Collector Resident Process Service ID. This must be identical to the ID in **service.dat**. | 32 characters | APRT_DC |
| 1 | **region_id** | Unique name for the tracking region. | 16 characters | ASMBLY LNE1_NOR |
| 2 | **region_desc** | Textual description of the region. | 40 characters | Assembly Line 1 Normal Region |
| 3 | **region_label** | Text to display with the region for user recognition. | 16 characters | AssmblyLne1 Nor |
| 4 | **tracking_type_id** | Primary tracking region type: Pool, Sequential or Shifting. | 16 characters | SEQUENTIAL |
| 5 | **region_cap** | Maximum number of available physical region locations. | Integer | 15 |
| 6 | **region_loc_cap** | Maximum number of items that can reside at a single region location. Be sure to include all serialized and non-serialized items. | Integer | 3 |
| 7 | **oos_cycle_count** | Number of items allowed to exit out of sequence from a tracking region. | Integer | 1 |
| 8 | **oos_error_code** | Indicates the action to be taken when an item does not arrive at the region exit transition indicator as expected. Use one of the following:<br>**DETAIN** – Move to detainment region.<br>**SCRAP** – Scrap (delete) item.<br>**DLY_DTN** – Allow item to be out-of-sequence using the **oos_cycle_ count** condition, then move to detainment region.<br>**DLY_SCRP** – Allow item to be out-of-sequence using **the oos_cycle_count** condition, then scrap (delete). | 8 characters | DLY_DTN |
| 9 | **GEF_reserved_1** | GE Fanuc Reserved. | 8 characters | Blank |
| 10 | **GEF_reserved_2** | GE Fanuc Reserved. | 8 characters | Blank |
| 11 | **GEF_reserved_3** | GE Fanuc Reserved. | 8 characters | Blank |
| 12 | **GEF_reserved_4** | GE Fanuc Reserved. | 8 characters | Blank |
| 13 | **det_region_id** | Unique name of the detainment region where items are sent when out of sequence. | 16 characters | DETAINMNT _A |

| No. | Field Name | Description | Data Type | Entry Example |
|---|---|---|---|---|
| 14 | **region_lockable** | Enable/disable (1/0) locked region, i.e. region must be locked to make modifications or to move items into the region. | One byte | 0 |
| 15 | **alarm_det_region** | Enable/disable (1/0) alarm generation when items are moved to the detainment region. | One byte | 1 |
| 16 | **alarm_region_not_empty** | Enable/disable (1/0) alarm generation when a disperse tracking region is not empty and new items have arrived in the region. | One byte | 0 |
| 17 | **alarm_region_lock** | Enable/disable (1/0) alarm generation when an item enters an IN-LOCKED region or exits an OUT-LOCKED region. | One byte | 0 |
| 18 | **alarm_loc_cap_excd** | Enable/disable (1/0) alarm generation when the number of items at a single region location exceeds the configured maximum. | One byte | 1 |
| 19 | **alarm_item_hold** | Enable/disable (1/0) alarm generation when an item enters a region with an attribute that matches an item-hold specification. | One byte | 1 |
| 20 | **alarm_unknown_item** | Enable/disable (1/0) alarm generation when information cannot be retrieved for a serialized item. | One byte | 1 |
| 21 | **enable_master_alarm_flag** | Enable/disable (1/0) alarming for this tracking region regardless of alarming enabled elsewhere. | One byte | 1 |
| 22 | **enable_master_log_flag** | Enable/disable (1/0) logging for this tracking region regardless of logging events enabled elsewhere. | One byte | 1 |
| 23 | **log_det_region** | Enable/disable (1/0) logging when items are moved to the detainment region. | One byte | 1 |
| 24 | **log_region_not_empty** | Enable/disable (1/0) logging when a disperse tracking region is not empty and new items have arrived in the region. | One byte | 0 |
| 25 | **log_region_stat** | Enable/disable (1/0) logging when an item enters an IN-LOCKED region or exits an OUT-LOCKED region. | One byte | 0 |
| 26 | **log_loc_cap_excd** | Enable/disable (1/0) logging when the number of items at a single region location exceeds the configured maximum. | One byte | 1 |
| 27 | **log_item_hold** | Enable/disable (1/0) logging when an item enters a region with an attribute that matches an item-hold specification. | One byte | 1 |
| 28 | **log_unknown_item** | Enable/disable (1/0) logging when information cannot be retrieved for a serialized item. | One byte | 1 |

| No. | Field Name | Description | Data Type | Entry Example |
|---|---|---|---|---|
| 29 | **log_item_scrapped** | Enable/disable (1/0) logging when an item has been scrapped (deleted) from the tracking region's queue. | One byte | 1 |
| 30 | **log_item_entry** | Enable/disable (1/0) logging whenever an item is entered into queue. | One byte | 1 |
| 31 | **log_item_exit** | Enable/disable (1/0) logging whenever an item exits the tracking queue. | One byte | 1 |
| 32 | **log_item_mod** | Enable/disable (1/0) logging whenever an item's tracking data is modified. | One byte | 0 |
| 33 | **log_prod_start** | Enable/disable (1/0) logging when tracking of an item starts in this region. | One byte | 0 |
| 34 | **log_prod_stop** | Enable/disable (1/0) logging when tracking of an item stops after this region. | One byte | 0 |
| 35 | **item_qty_ptid** | Point ID that displays the total number of items contained in the region; must be of type INT or UINT. | 55 characters | ASMBLYLNE1_TOTAL |
| 36 | **stat_ptid** | Point ID that represents the region's status; must be of type DIUNT. | 55 characters | ASMBLYLNE1_STAT |
| 37 | **region_type** | Secondary tracking region type. Use the following codes:<br>1 = Normal<br>2 = Combine<br>3 = Disperse<br>Note: If a Combine or Disperse region, **region_cap** field must be set to 1, and **region_loc_cap** must be greater than 1. | Integer | 1 |
| 38 | **auto_assoc_region** | Indicates if items are automatically associated in this region; must be of secondary region type Combine. Yes/No (1/0). | One byte | 0 |
| 39 | **resource_id** | Uniquely identifies the resource against which the alarm will be generated at the tracking region level.<br>Note: If a resource is not designated here, the alarm will be generated against the default, "$SYSTEM." | 16 characters | OpWks_1 |
| 40 | **cfg_override** | GE Fanuc Reserved. | One byte | Blank |
| 41 | **alarm_oos_region** | Enable/disable (1/0) alarm generation when an "out-of-sequence" condition occurs. | One byte | 1 |
| 42 | **alarm_inval_type** | Enable/disable (1/0) alarm generation when an invalid item type enters the region. | One byte | 1 |
| 43 | **log_inval_type** | Enable/disable (1/0) logging when an invalid item type enters the region. | One byte | 1 |

| No. | Field Name | Description | Data Type | Entry Example |
|---|---|---|---|---|
| 44 | **alarm_unknown_type** | Enable/disable (1/0) alarm generation when an unknown item type enters the region. | One byte | 1 |
| 45 | **log_unknown_type** | Enable/disable (1/0) logging when an unknown item type enters the region. | One byte | 1 |
| 46 | **alarm_reg_cap** | Enable/disable (1/0) alarm generation when the region's configured capacity has been exceeded. | One byte | 0 |
| 47 | **log_reg_cap** | Enable/disable (1/0) logging when the region's configured capacity has been exceeded. | One byte | 0 |
| 48 | **auto_move_point** | Text Point ID that displays when an item moved into the region due to a "Point-based" move. The Point contains the item ID and optional information depending on the value of the **num_setpt_param** set in the global parameters files. | 55 characters | Blank |
| 49 | **tran_region_id** | Indicates the transition region ID for a shifting region. | 16 characters | Blank |

### *Configuration File Example*


prt_region.idt - Notepad
File Edit Search Help
* 40 cfg_override GE/FANUC Reserved; cfg overrides glb mem
* 41 alarm_oos_region en/dis oos condition alrm
* 42 alarm_inval_type en/dis invalid item type alrm
* 43 log_inval_type en/dis invalid item type logging
* 44 alarm_unknown_type en/dis unknown item type alrm
* 45 log_unknown_type en/dis unknown item type logging
* 46 alarm_reg_cap en/dis region capacity alrm
* 47 log_reg_cap en/dis region capacity logging
* 48 auto_move_point automatic item move setpoint
* 49 tran_region_id Tracking Region's Transition Region ID
APRT_DC|ASMBLYLNE1_NOR|Assembly Line 1 Normal Region|AsmblyLne1 Nor|SEQUENTIAL|15|-
3|1|DLY_DTN|||||DETAINMNT_A|0|1|0|0|1|1|1|1|1|1|0|0|1|1|1|1|1|1|0|0|0|-
ASMBLYLNE1_TOTAL|ASMBLYLNE1_STAT|1|0|OpWks_1|||1|1|1|1|1|0|0||
APRT_DC|ROBOT1_COM|Robot 1 Combine Region|Robot1 Com|SEQUENTIAL|1|-
4|0|||||||0|1|0|0|1|1|1|1|1|1|0|0|1|1|1|1|1|1|0|0|0|-
ROBOT1_TOTAL|ROBOT1_STAT|2|1|RbtDevice 1|||1|1|1|1|1|0|0||
2 records created.
APRT_DC|ASMBLYLNE1_NOR|Assembly Line 1 Normal Region|AsmblyLne1 Nor|SEQUENTIAL|15|-
3|1|DLY_DTN | | | | |DETAINMNT_A|0|1|0|0|1|1|1|1|1|1|0|0|1|1|1|1|1|1|0|0|0|-
ASMBLYLNE1_TOTAL AMBLYLNE1_STAT|1|0|OpWks_1| |1|1|1|1|1|0|0| |
Expanded view of the first record.
Last field blank, no delimiter needed after.
Continuation character appends next line to record.

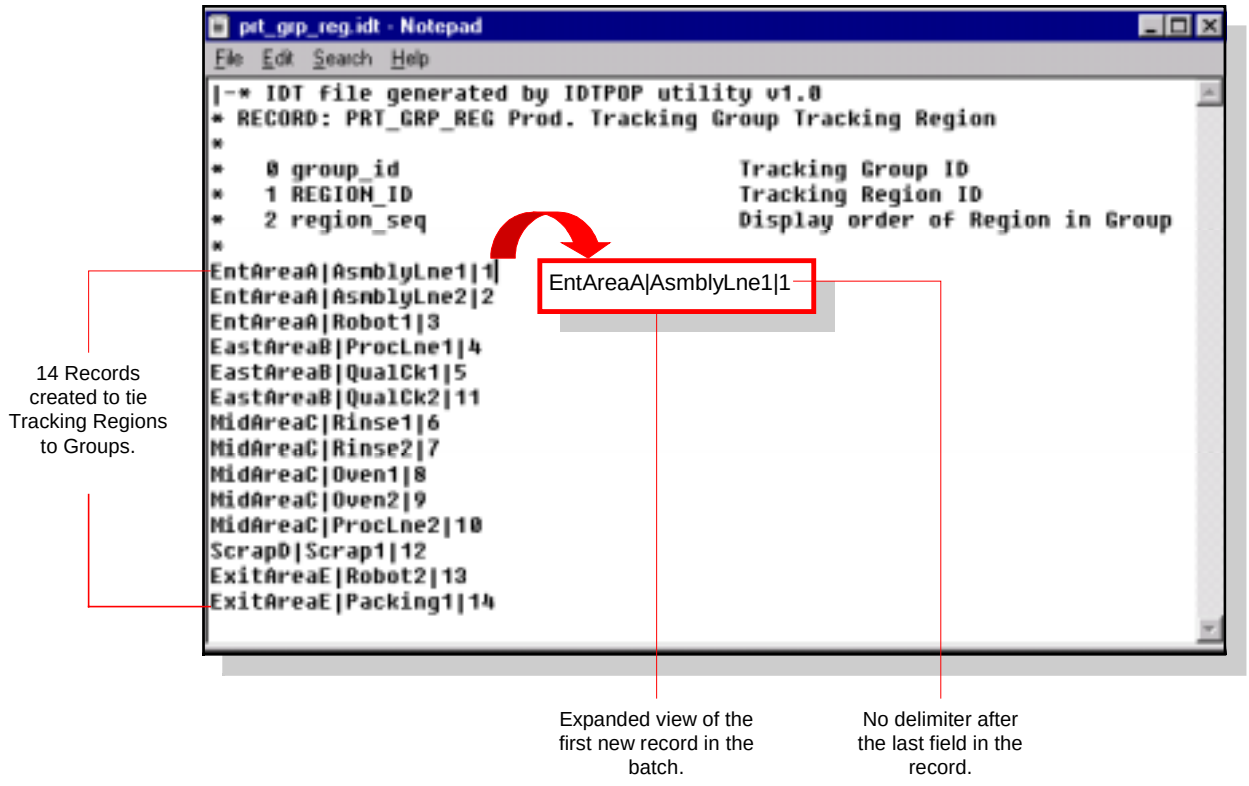
# PRT_GRP_REG

The PRT Group Tracking Region (**prt_grp_reg**) configuration file is used to tie Regions to Groups. Each of the fields is described below, complete with entry examples and a sample configuration file.

**REQUIRED FILE**

| No. | Field Name | Description | Data Type | Entry Example |
|---|---|---|---|---|
| 0 | **group_id** | Unique name for the Group responsible for the corresponding region (see Field 1). | 16 characters | EntAreaA |
| 1 | **region_id** | Unique name for the region in the Group. | 16 characters | AsmblyLne1 |
| 2 | **region_seq** | Sequence number of the region in this Group. Optionally, use this field to represent the physical order of the regions in a production line or in the group (used for display purposes only). | Integer | 1 |

### *Configuration File Example*



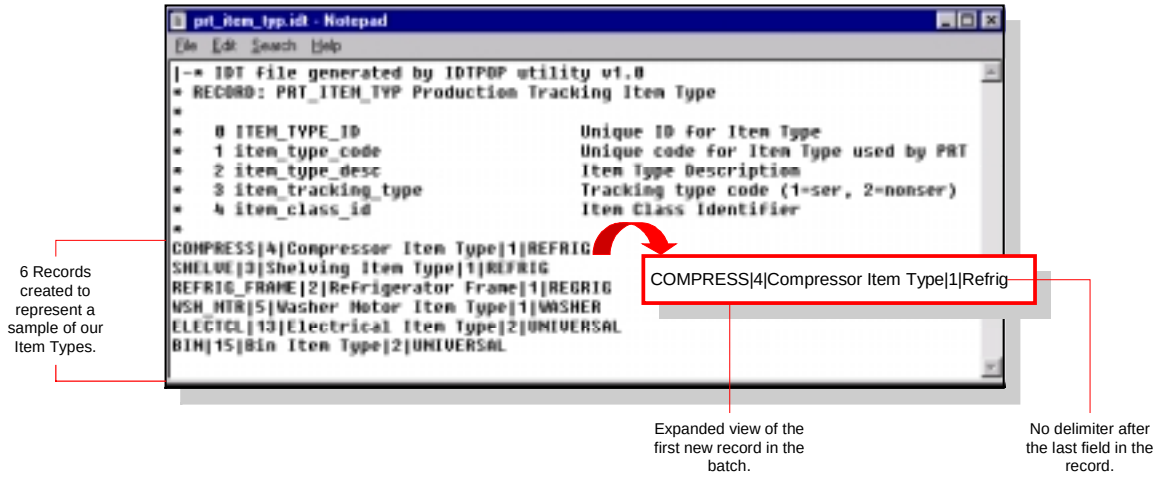
# PRT_ITEM_TYP

The PRT Item Type (**prt_item_typ**) configuration file is used to define all of the item types that the Tracking Model is responsible for monitoring. Each of the fields is described below, complete with entry examples and a sample configuration file.

**REQUIRED FILE**

| No. | Field Name | Description | Data Type | Entry Example |
|---|---|---|---|---|
| 0 | **item_type_id** | Unique name for the item type. | 16 characters | COMPRESS |
| 1 | **item_type_code** | Unique code identifying the item type to be used internally by PRT. | Integer | 4 |
| 2 | **item_type_desc** | Textual description of the item type. | 40 characters | Compressor Item Type |
| 3 | **item_tracking _type** | Code identifying the tracking type: 1 = Serialized; 2 = Non-Serialized. | One byte | 1 |
| 4 | **item_class_id** | Name of the item class to which this item type belongs. **Reminder:** Item Classes are not required but are recommended. Feel free to create one Item Class for all of your item types. | 16 characters | Refrig |

## *Configuration File Example*


prt_item_typ.idt - Notepad
File Edit Search Help
|-* IDT File generated by IDTPDP utility v1.0
* RECORD: PRT_ITEM_TYP Production Tracking Item Type
*
* 0 ITEM_TYPE_ID Unique ID for Item Type
* 1 item_type_code Unique code for Item Type used by PRT
* 2 item_type_desc Item Type Description
* 3 item_tracking_type Tracking type code (1=ser, 2=nonser)
* 4 item_class_id Item Class Identifier
*
COMPRESS|4|Compressor Item Type|1|REFRIG
SHELVE|3|Shelving Item Type|1|REFRIG
REFRIG_FRAME|2|Refrigerator Frame|1|REGRIG
WSH_MTR|5|Washer Motor Item Type|1|WASHER
ELECTCL|13|Electrical Item Type|2|UNIVERSAL
BIN|15|Bin Item Type|2|UNIVERSAL
COMPRESS|4|Compressor Item Type|1|Refrig
6 Records created to represent a sample of our Item Types.
Expanded view of the first new record in the batch.
No delimiter after the last field in the record.

# PRT_ROUTE

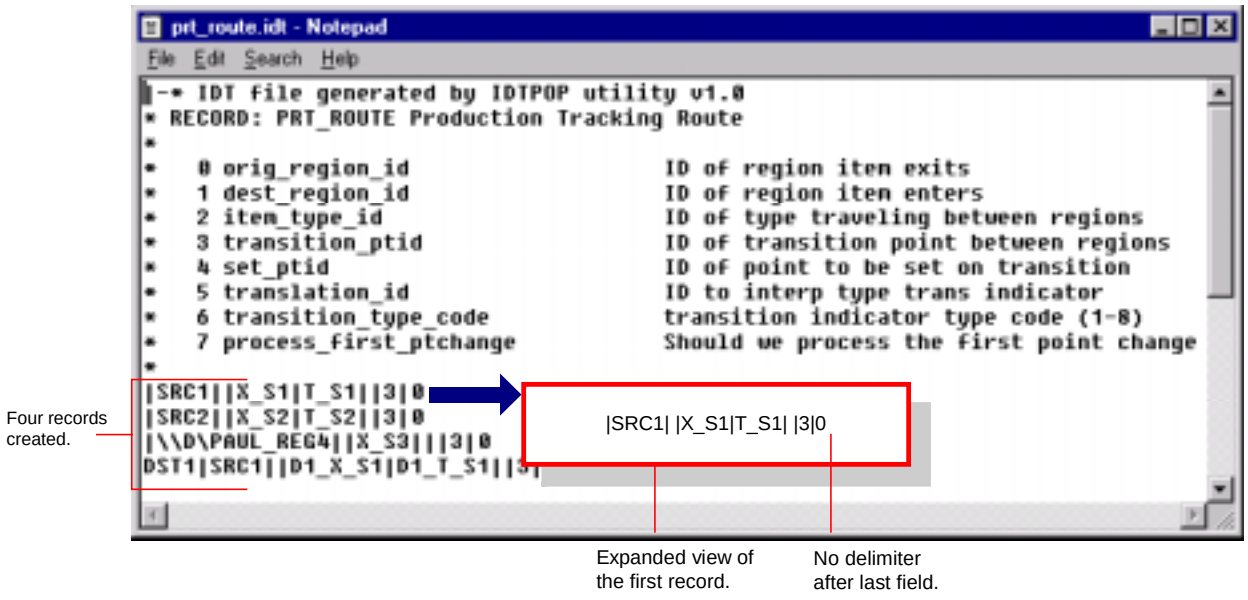
The PRT Route (**prt_route**) configuration file is used to define all of the routes that exist between regions. This enables automatic transition of items based on changes in CIMPLICITY Point values. Each of the fields is described below, complete with entry examples and a sample configuration file.

**REQUIRED FILE**

| No. | Field Name | Description | Data Type | Entry Example |
|---|---|---|---|---|
| 0 | **orig_region_id** | Name of the source region (where the item will exit). | 16 characters | NULL |
| 1 | **dest_region_id** | Name of the destination region (where the item will enter). | 16 characters | SRC1 |
| 2 | **item_type_id** | Unique name for the item type that may travel between the two regions. Use the wildcard character (*) to validate all item types through this route. | 16 characters | NULL |
| 3 | **transition_ptid** | Point ID that represents the transition between the two regions. | 55 characters | X_S1 |
| 4 | **set_ptid** | Point ID that will display when the item is transitioned. Must be of class, Text and should be no more than 36 characters. Data is displayed as follows:<br>16 characters – Item Type ID.<br>20 characters – Item ID when item type is serialized; otherwise, associated or parent item ID. | 55 characters | T_S1 |
| 5 | **translation_id** | Depending on Field 6 (**translation_type_code**), configure as follows:<br>translation_type_code = 2: translation to use to interpret this transition indicator.<br>translation_type_code = 3, 4 or 5: Up to 3 characters that will be attached as a prefix to the Point value prior to processing. This allows items with the same ID to be identified in different parts of the factory.<br>translation_type_code = 1, 6 or 7: NOT USED. | 16 characters | NULL |

| No. | Field Name | Description | Data Type | Entry Example |
|---|---|---|---|---|
| 6 | **translation_type_code** | Unique code to represent the tracking indicator type as follows:<br>**1** = Limit switch<br>**2** = Type detector (integer Point)<br>**3** = Automatic Item ID displaying Item ID (Text Point)<br>**4** = Automatic Item ID displaying Reference ID (Text Point)<br>**5** = Associated Reference ID to Item (Text Point)<br>**6** = Associated Item Type to Item (Integer Point)<br>**7** = Positive Edge Trigger (transition occurs when value changes from zero to nonzero)<br>**8** = Negative Edge Trigger (transition occurs when value changes from nonzero to zero) | One byte | 3 |
| 7 | **Process_first_ptchange** | ▪ FALSE or undefined transition point ignores the first point change of a route.<br>▪ TRUE processes the first point change as a transition point. | 0=FALSE<br>1=TRUE | 0 |

### *Configuration File Example*


prt_route.idt - Notepad
File Edit Search Help
|-* IDT file generated by IDTPOP utility v1.0
* RECORD: PRT_ROUTE Production Tracking Route
*
* 0 orig_region_id ID of region iten exits
* 1 dest_region_id ID of region iten enters
* 2 item_type_id ID of type traveling between regions
* 3 transition_ptid ID of transition point between regions
* 4 set_ptid ID of point to be set on transition
* 5 translation_id ID to interp type trans indicator
* 6 transition_type_code transition indicator type code (1-8)
* 7 process_first_ptchange Should we process the first point change
*
|SRC1||X_S1|T_S1||3|0
|SRC2||X_S2|T_S2||3|0
|\\D\PAUL_REG4||X_S3|||3|0
DST1|SRC1||D1_X_S1|D1_T_S1||3|
|SRC1| |X_S1|T_S1| |3|0
Four records created.
Expanded view of the first record.
No delimiter after last field.

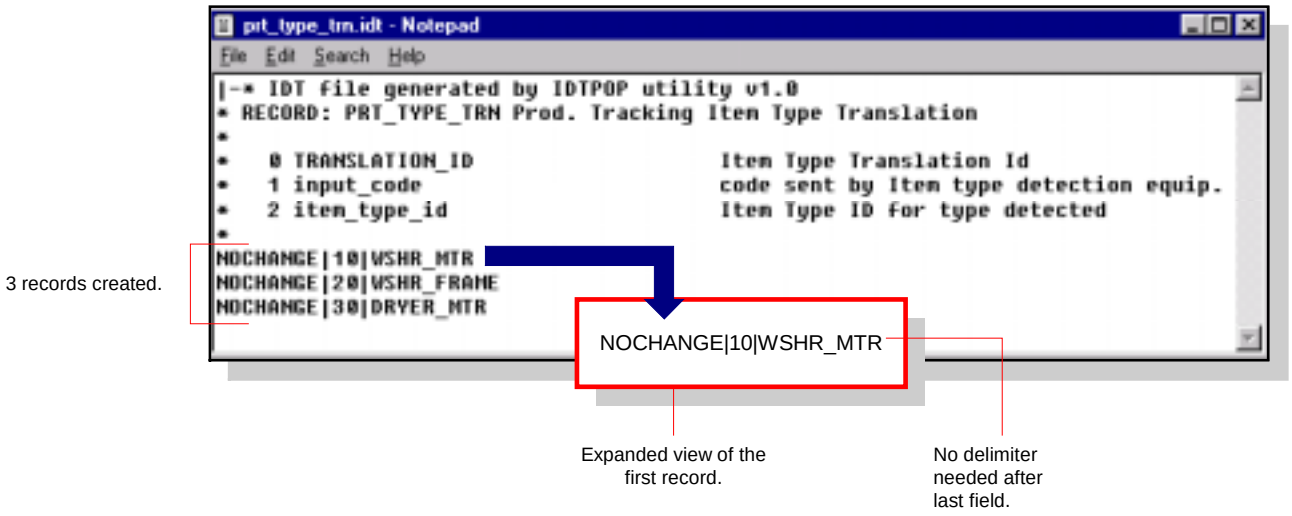
# PRT_TYPE_TRN

The PRT Item Type Translation (**prt_type_trn**) configuration file is used to translate item type codes received from detection equipment into item type IDs. Once translated, Tracker recognizes the IDs via the PRT database. Configure this file when your **prt_route** records have a translation type code of 2. Each of the fields is described below, complete with entry examples and a sample configuration file.

**OPTIONAL FILE**

| No. | Field Name | Description | Data Type | Entry Example |
|---|---|---|---|---|
| 0 | **translation_id** | Unique name of the item type translation. | 16 characters | NOCHANGE |
| 1 | **input_code** | Unique code received from detection equipment signaling the item type. | Integer | 10 |
| 2 | **item_type_id** | Unique name of the item type that is to be translated using the input code from the detection equipment. | 16 characters | WSHR_ MOTOR |

### *Configuration File Example*


prt_type_trn.idt - Notepad
File Edit Search Help
|-* IDT file generated by IDTPOP utility v1.0
* RECORD: PRT_TYPE_TRN Prod. Tracking Item Type Translation
*
* 0 TRANSLATION_ID Item Type Translation Id
* 1 input_code code sent by Item type detection equip.
* 2 item_type_id Item Type ID for type detected
*
NOCHANGE|10|WSHR_MTR
NOCHANGE|20|WSHR_FRAME
NOCHANGE|30|DRYER_MTR
3 records created.
NOCHANGE|10|WSHR_MTR
Expanded view of the first record.
No delimiter needed after last field.

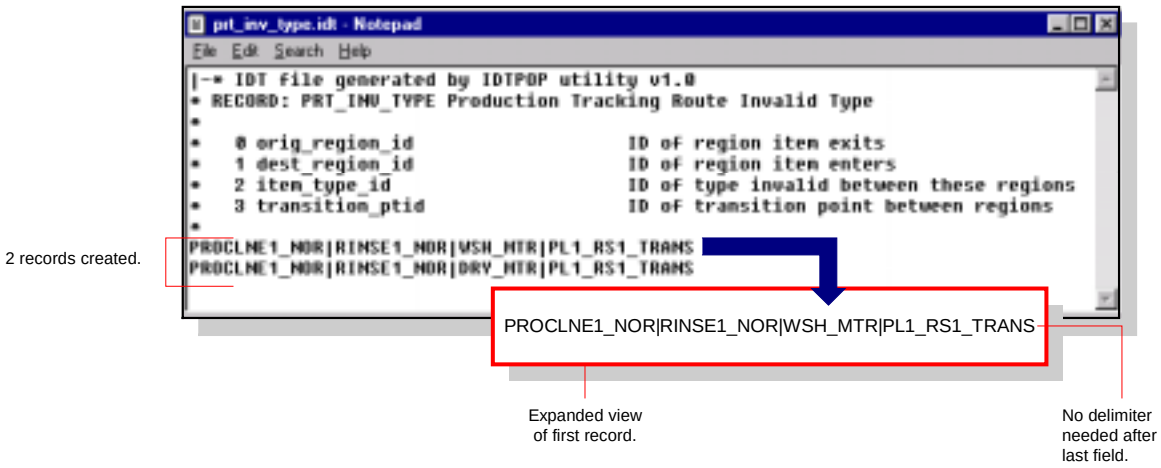
# PRT_INV_TYPE

The PRT Invalid Type (**prt_inv_type**) configuration file is used to specify when a configured item type is not valid for a tracking route. Recall that you can use the wildcard character in the **prt_route** configuration file to identify the item types. When the wildcard character is used, you can use this file to eliminate the invalid item types from the route. Each of the fields is described below, complete with entry examples and a sample configuration file.

**OPTIONAL FILE**

| No. | Field Name | Description | Data Type | Entry Example |
|---|---|---|---|---|
| 0 | **orig_region_id** | Unique name of the region from which the item type will exit. | 16 characters | PROCLNE1 _NOR |
| 1 | **dest_region_id** | Unique name of the region where the item type may enter. | 16 characters | RINSE1_NOR |
| 2 | **item_type_id** | Unique name of the item type that is not allowed to travel between the regions; in other words, the invalid item type. | 16 characters | WSH_MTR |
| 3 | **transition_ptid** | Point ID that represents the transition between the two regions. | 55 characters | PL1_RS1 _TRANS |

### *Configuration File Example*


prt_inv_type.idt - Notepad
File Edit Search Help
|-* IDT file generated by IDTPOP utility v1.0
* RECORD: PRT_INV_TYPE Production Tracking Route Invalid Type
*
* 0 orig_region_id ID of region item exits
* 1 dest_region_id ID of region item enters
* 2 item_type_id ID of type invalid between these regions
* 3 transition_ptid ID of transition point between regions
*
PROCLNE1_NOR|RINSE1_NOR|WSH_MTR|PL1_RS1_TRANS
PROCLNE1_NOR|RINSE1_NOR|DRY_MTR|PL1_RS1_TRANS
2 records created.
PROCLNE1_NOR|RINSE1_NOR|WSH_MTR|PL1_RS1_TRANS
Expanded view of first record.
No delimiter needed after last field.

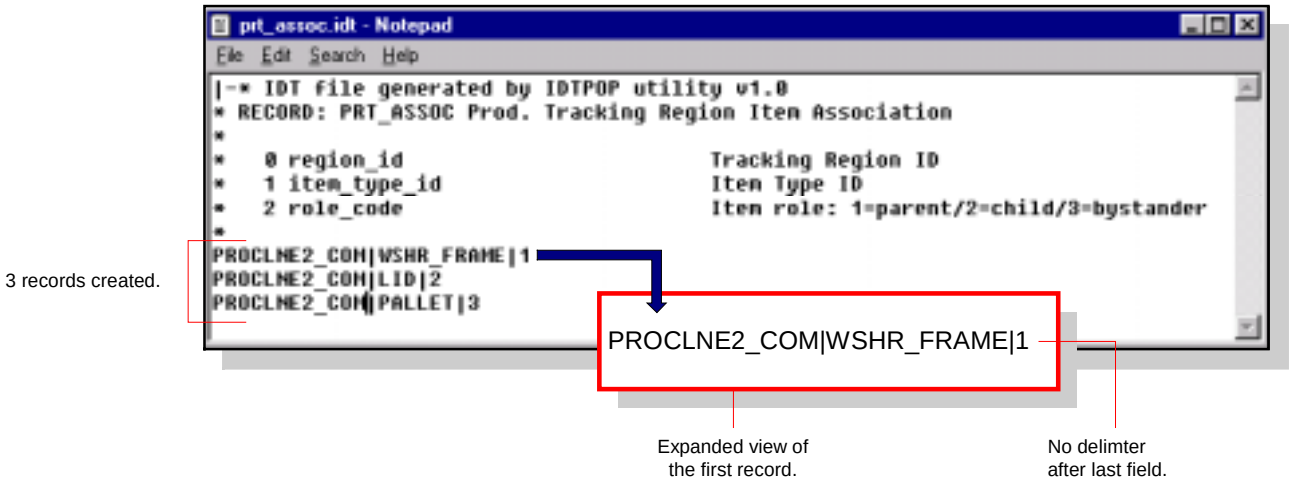
# PRT_ASSOC

The PRT Item Association (**prt_assoc**) configuration file is used to define a parent-child association between a serialized and non-serialized item. Configure one record for every parent-child association to be made during production. Each of the fields is described below, complete with entry examples and a sample configuration file.

**OPTIONAL FILE**

| No. | Field Name | Description | Data Type | Entry Example |
|---|---|---|---|---|
| 0 | **region_id** | Name of the region in which the parent-child association will take place. | 16 characters | PROCLNE2 _COM |
| 1 | **item_type_id** | Unique name of the item type that will enter the above region (can be serialized or non-serialized). | 16 characters | WSHR_FRAME |
| 2 | **role-code** | Code indicating the role of the item type as follows:<br>1 = Parent item (serialized)<br>2 = Child item (non-serialized item),<br>3 = Bystander (no association will take place) | One byte | 1 |

### *Configuration File Example*



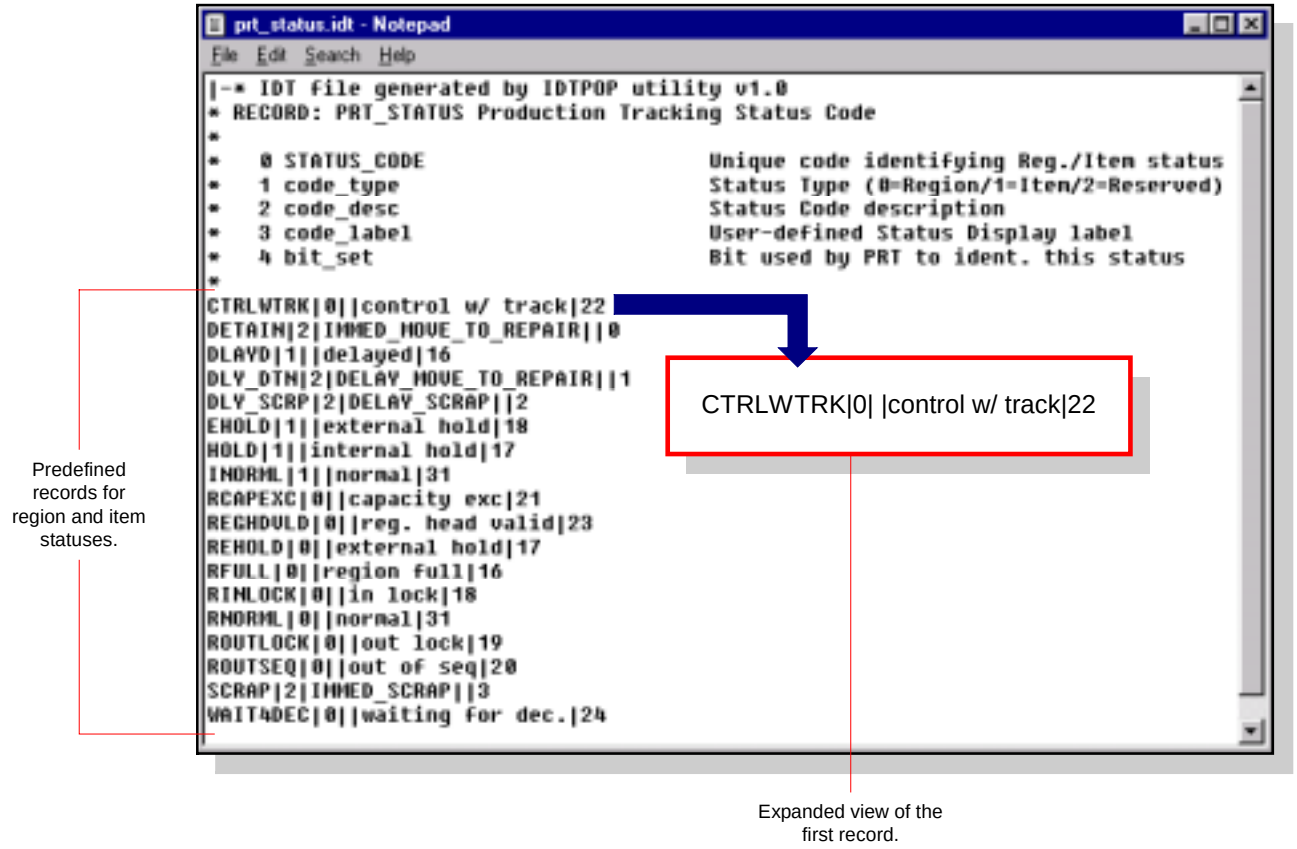
# PRT_STATUS

The PRT Status (**prt_status**) configuration file is used to assign a status code for each valid region status; for example, capacity, sequential error, invalid or unknown type, etc. Similarly, a record is configured for each valid item status; for example, hold, no-hold, delayed, etc. A number of statuses have predefined records, and you may configure additional records to define other status types. Each of the fields is described below, complete with entry examples and a sample configuration file.

**OPTIONAL FILE**

| No. | Field Name | Description | Data Type | Entry Example |
|---|---|---|---|---|
| 0 | **status_code** | Unique code to identify the status of the region or item. | 8 characters | CTRLWTRK |
| 1 | **code_type** | Unique code representing what you are defining a status code for, as follows:<br>0 = Region Status Code<br>1 = Item Status Code<br>2 = Reserved | One byte | 0 |
| 2 | **code_desc** | Textual description of the status code. | 40 characters | Blank |
| 3 | **code_label** | Text to display with the status code for user recognition. Must be unique. | 16 characters | Control w/track |
| 4 | **bit_set** | Bit to be set by PRT that will represent this status code. | Long word | 22 |

### *Configuration File Example*



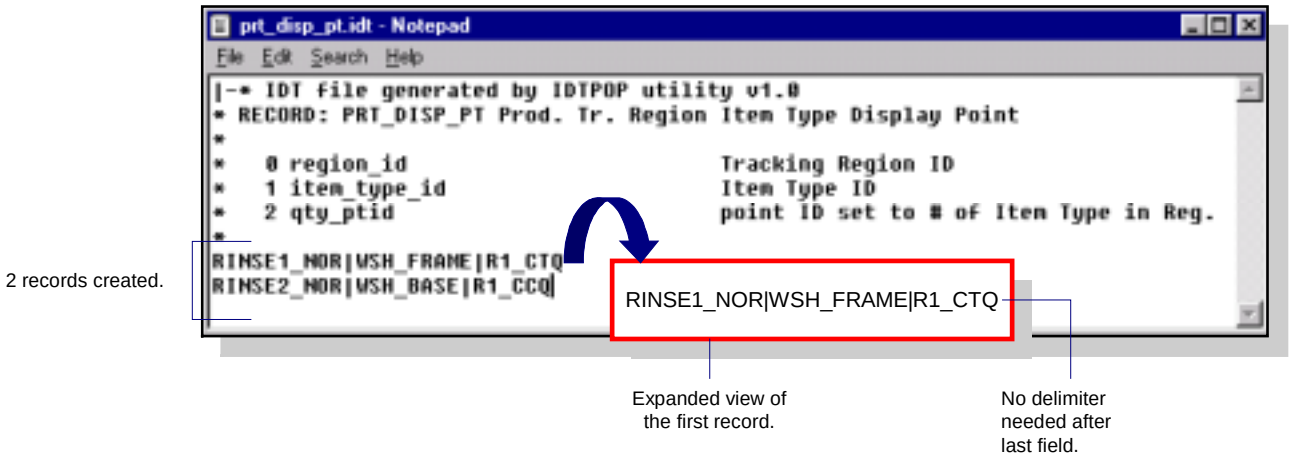
# PRT_DISP_PT

The PRT Display Point (**prt_disp_pt**) configuration file is used to define the Item Type quantity in a region to be displayed through a setpoint. This file must be configured when **prt_grd** is configured. Each of the fields is described below, complete with entry examples and a sample configuration file.

**OPTIONAL FILE**

| No. | Field Name | Description | Data Type | Entry Example |
|---|---|---|---|---|
| 0 | **region_id** | Unique name for the tracking region. | 16 characters | RINSE1_NOR |
| 1 | **item_type_id** | Unique name for the item type. | 16 characters | WSH_FRAME |
| 2 | **qty_ptid** | Unique name for the setpoint to display the item type quantity in the region. | 55 characters | R1_CTQ |

## *Configuration File Example*


prt_disp_pt.idt - Notepad
File Edit Search Help
|-* IDT file generated by IDTPOP utility v1.0
* RECORD: PRT_DISP_PT Prod. Tr. Region Item Type Display Point
*
* 0 region_id Tracking Region ID
* 1 item_type_id Item Type ID
* 2 qty_ptid point ID set to # of Item Type in Reg.
*
RINSE1_NOR|WSH_FRAME|R1_CTQ
RINSE2_NOR|WSH_BASE|R1_CCQ
2 records created.
RINSE1_NOR|WSH_FRAME|R1_CTQ
Expanded view of the first record.
No delimiter needed after last field.

# PRT_ATT_PT

The PRT Region Item Attribute Display Point (**prt_att_pt**) configuration file is used to define the setpoint ID that will hold the value for an item attribute in a region location. This file must be configured when **prt_grd** is configured. Each of the fields is described below, complete with entry examples and a sample configuration file.

**OPTIONAL FILE**

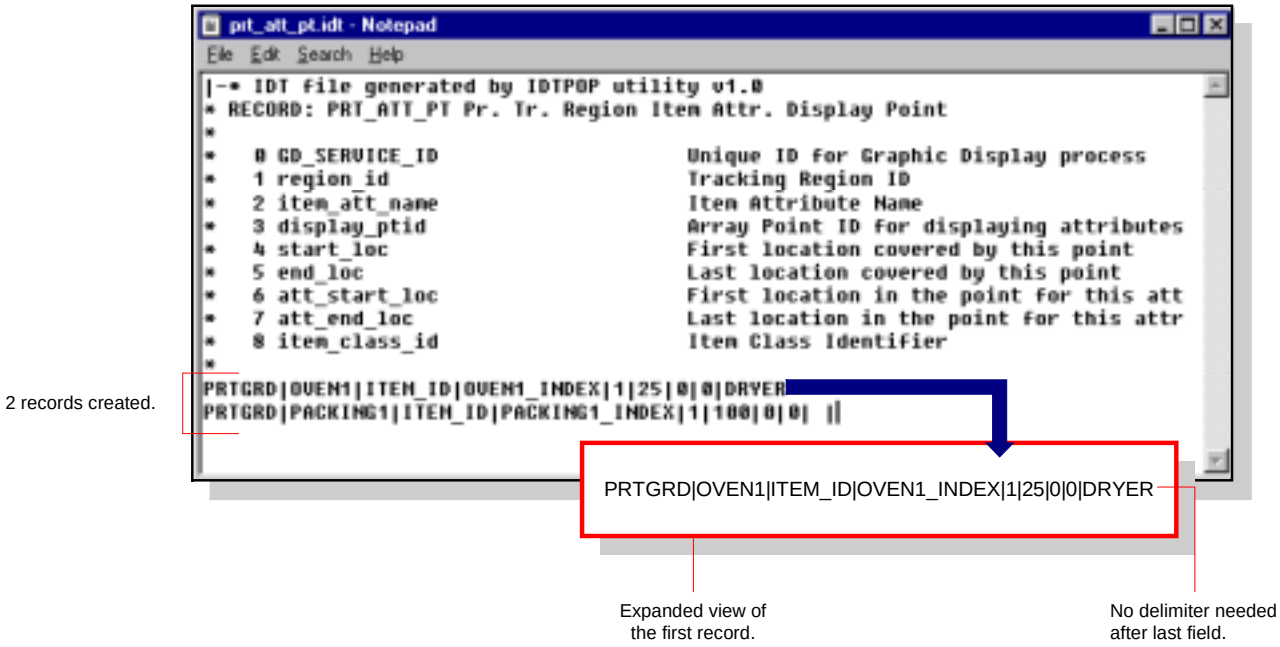
| No. | Field Name | Description | Data Type | Entry Example |
|---|---|---|---|---|
| 0 | **gd_service_id** | Unique name of the Process Service ID. | 32 characters | PRTGRD |
| 1 | **region_id** | Unique name for the region. | 16 characters | OVEN1 |
| 2 | **item_att_name** | Textual description of the item attribute, as follows:<br>**PRT_ITEM_ID**<br>**PRT_REFERENCE_ID**<br>**PRT_ITEM_TYPE_ID**<br>**PRT_ITEM_STATUS**<br>**PRT_ASSOC_ITEM**<br>Note: When using **prt_item_status** in this field, the values in **att_start_loc** and **att_end_loc** must both be 0. | 16 characters | ITEM_ID |
| 3 | **display_ptid** | Unique name of the array point. The definition of the point depends on the **item_att_name** field, as follows:<br>**PRT_ITEM_ID** – Point class must be Text; Point type must define a text size of at least 20.<br>**PRT_REFERENCE_ID** – Point class must be Text; Point type must define a text size of at least 20.<br>**PRT_ITEM_TYPE_ID** – Point class must be Text; Point type must define a text size of at least 20.<br>**PRT_ITEM_STATUS** – Point class must be Analog; Point type must be ANALOG_U32.<br>**PRT_ASSOC** – Point class must be Text; Point type must define a text size of at least 20.<br>User defined – Point class must be Text. The number of region locations covered by the Point determines the number of elements needed. Use this formula to calculate the number of elements:<br>(**end_loc**) – (**start_loc**) + 1 | 55 characters | OVEN1_ INDEX |

| No. | Field Name | Description | Data Type | Entry Example |
|---|---|---|---|---|
| 4 | **start_loc** | First location in the region covered by this Point.<br>If this value is '-2', then the Point displays data from the last <n> items in the region, where <n> is the number of elements in the Point. | Integer | 1 |
| 5 | **end_loc** | Last location in the region covered by this Point.<br>Due to a restriction on Point size, one Point may not be able to cover an entire region; in which case, the region must be broken into subsets. Each subset as defined by a starting and ending location, is covered by a single Point. | Integer | 25 |
| 6 | **att_start_loc** | First location in the attribute to parse when setting value of **display_ptid**. The default value is 0.<br>*See Note.* | Integer | 0 |
| 7 | **att_end_loc** | Last location in the attribute to parse when setting value of **display_ptid**. The default value is 0.<br>*See Note.* | Integer | 0 |
| 8 | **item_class_id** | Name of the item class that can provide values for this Point. If a class is not specified, any item type will provide data. If more than one item matches the item filtering, the value of the Point is indeterminate. | 16 characters | DRYER |

**Note:** The value of (**att_end_loc** – **att_start_loc** + 1) can be no greater than the data length of **display_ptid**.

When using the default value (0) in either the **att_end_loc** or the **att_start_loc** field, the entire attribute will be transferred to **display_ptid** and no parsing will take place.

### *Configuration File Example*


prt_att_pt.idt - Notepad
File Edit Search Help
|-* IDT file generated by IDTPOP utility v1.0
* RECORD: PRT_ATT_PT Pr. Tr. Region Item Attr. Display Point
*
* 0 GD_SERVICE_ID Unique ID for Graphic Display process
* 1 region_id Tracking Region ID
* 2 item_att_name Item Attribute Name
* 3 display_ptid Array Point ID for displaying attributes
* 4 start_loc First location covered by this point
* 5 end_loc Last location covered by this point
* 6 att_start_loc First location in the point for this att
* 7 att_end_loc Last location in the point for this attr
* 8 item_class_id Item Class Identifier
*
PRTGRD|OVEN1|ITEM_ID|OVEN1_INDEX|1|25|0|0|DRYER
PRTGRD|PACKING1|ITEM_ID|PACKING1_INDEX|1|100|0|0|
2 records created.
PRTGRD|OVEN1|ITEM_ID|OVEN1_INDEX|1|25|0|0|DRYER
Expanded view of the first record.
No delimiter needed after last field.

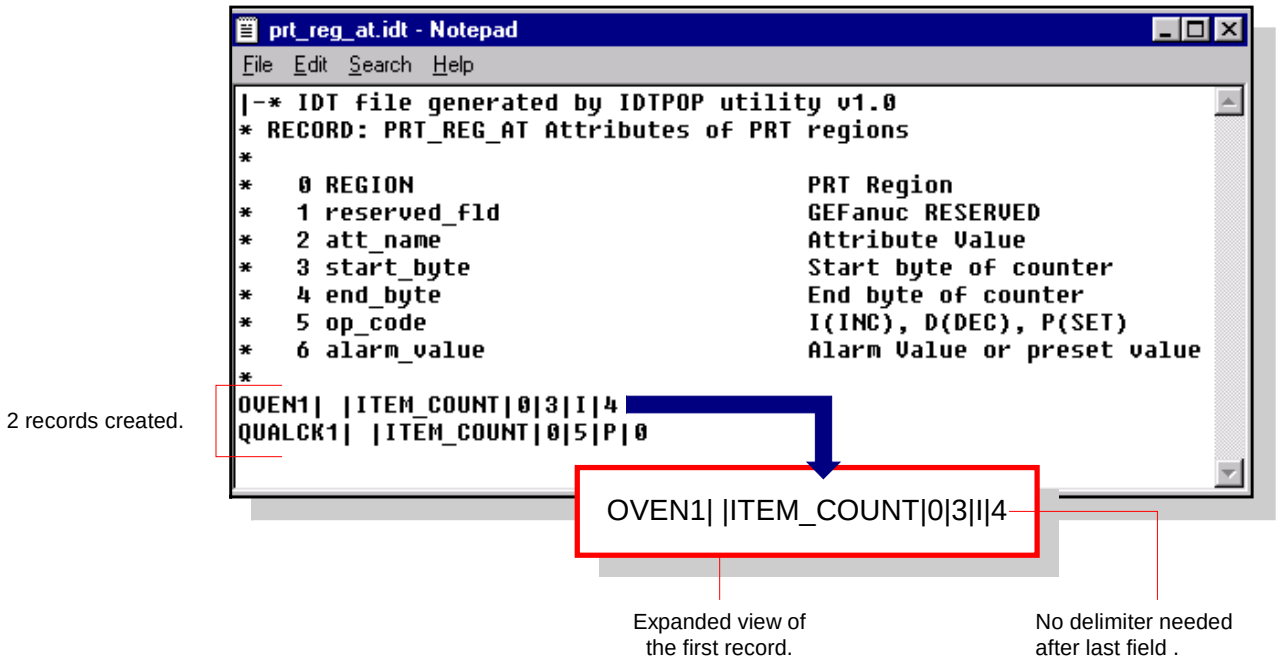
# PRT_REG_AT

The PRT Region Attribute Counter Definitions (**prt_reg_at**) configuration file is used to define counters in item attributes. This will in turn trigger the action taken when an item containing the defined attribute enters the region. Each of the fields is described below, complete with entry examples and a sample configuration file.

**OPTIONAL FILE**

| No. | Field Name | Description | Data Type | Entry Example |
|---|---|---|---|---|
| 0 | **region** | Name of the region where the operation will take place | 16 characters | OVEN1 |
| 1 | **reserved_fld** | GE Fanuc Reserved. | 16 characters | Blank |
| 2 | **att_name** | Name of the attribute containing the counter. | 16 characters | ITEM_COUNT |
| 3 | **start_byte** | Index of the first character of the count within the attribute, i.e. 1-16.<br>A value of 0 indicates use of the entire attribute. | Long word | 0 |
| 4 | **end_byte** | Index of the last character of the count within the attribute, i.e. 1-16. | Long word | 3 |
| 5 | **op_code** | Operation type code that is to impact the counter, as follows:<br>I = Increment<br>D = Decrease<br>P = Preset | One character | I |
| 6 | **alarm_value** | Enter the value that will generate an alarm based on the **op_code** field, as follows:<br>I (Increment) = over value will generate alarm.<br>D (Decrease) = under value will generate alarm.<br>P (Preset) = value is equal to the preset value. | Long word | 4 |

### *Configuration File Example*


prt_reg_at.idt - Notepad
File Edit Search Help
|-* IDT file generated by IDTPOP utility v1.0
* RECORD: PRT_REG_AT Attributes of PRT regions
*
*  0 REGION  PRT Region
*  1 reserved_fld  GEFanuc RESERVED
*  2 att_name  Attribute Value
*  3 start_byte  Start byte of counter
*  4 end_byte  End byte of counter
*  5 op_code  I(INC), D(DEC), P(SET)
*  6 alarm_value  Alarm Value or preset value
*
OVEN1| |ITEM_COUNT|0|3|I|4
QUALCK1| |ITEM_COUNT|0|5|P|0
2 records created.
OVEN1| |ITEM_COUNT|0|3|I|4
Expanded view of the first record.
No delimiter needed after last field .

# PRT_ATTRIBUTES.CFG

The PRT Attribute Labels (**prt_attributes.cfg**) configuration file is used to assign attributes to your items. Attributes can be assigned to all item types or can be assigned to a specific item class. When assigning attributes to an item class, use the keyword ITEMCLASS. For each PRT Service that you have designated in your Tracking Model, you will need a separate configuration file named for the Service.

This file can be viewed and edited without the IDTPOP Utility. Simply enter **notepad prt_attributes.cfg** (where **Notepad** is the text editor) on the Command line in the MSDos prompt window. This file is not required to configure your Tracking Model, but is recommended in order to gather comprehensive data on the items tracked through production.

**OPTIONAL FILE**

**To assign attributes:**

1. Assign Attributes for all items as follows:

   **Note:** Enter attributes that you want to assign for all of your items in a single record. Be sure to use the delimiter character to separate attributes.

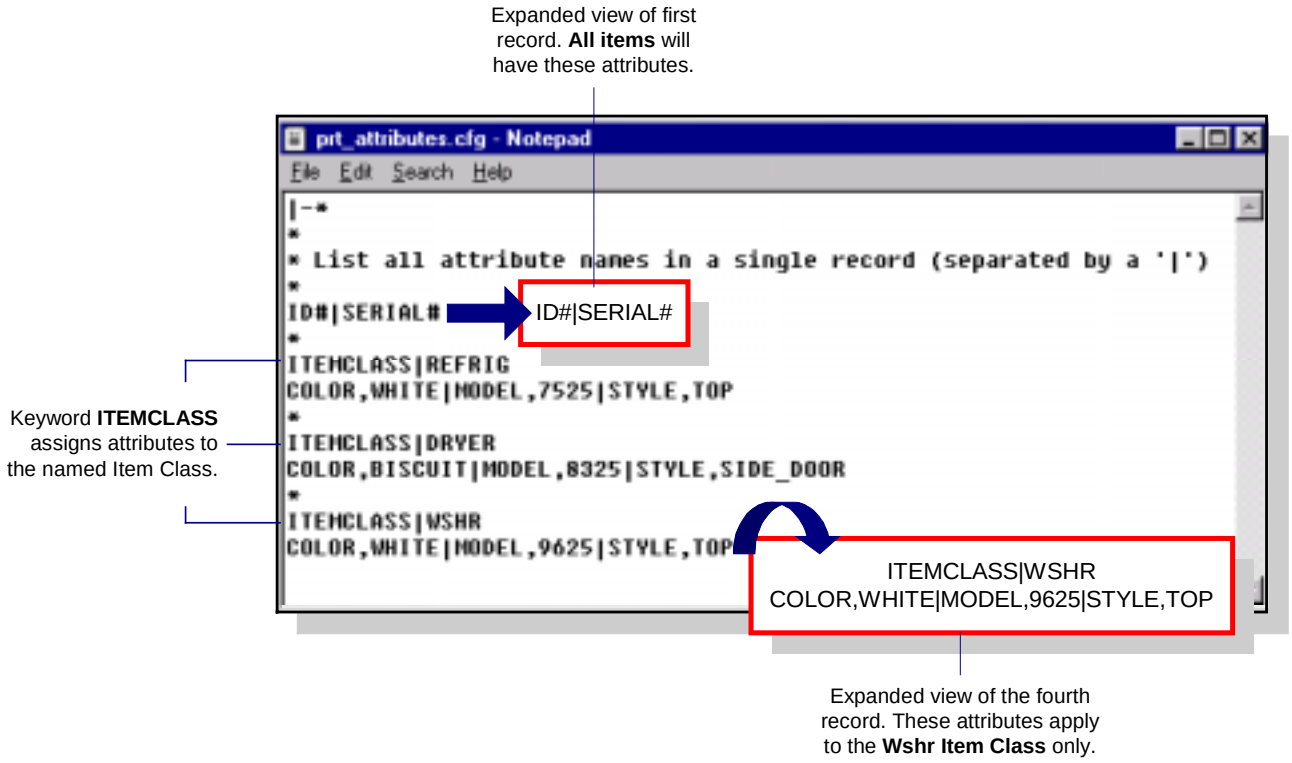
2. Assign Attributes for an item class(es) as follows:
   A. Start a new record.
   B. Enter the keyword ITEMCLASS.
   C. Enter the delimiter character.
   D. Enter the name of the Item Class.
   E. Press **ENTER**.
   F. Enter the attributes that you want to assign for the Item Class in a single record. Be sure to use the delimiter character to separate attributes.

3. Click File on the menu bar once you have assigned attributes for your items and proceed as follows:

| Do | When |
|---|---|
| Select Save | The PRT_UI is configured to view all PRT Services. |
| Select Save As | The PRT_UI is configured to view a specific service<br>*Then*<br>A. Enter **aprt_window.cfg** in the File Name field,<br>*Where*<br>**aprt** is the name of the PRT Service<br>B. Click **Save.** |

4. Close the text editor.

## *Configuration File Example*



# PRT_STND.CFG

The PRT Standby Process (**prt_stnd.cfg**) configuration file is used to assign a PRT Server to the PRT Standby Process. The PRT Standby Process will produce a PRT import file in the local log directory that contains all the contents of the PRT Server when contact was lost. This file is not provided with Tracker and must be manually configured.

**OPTIONAL FILE**

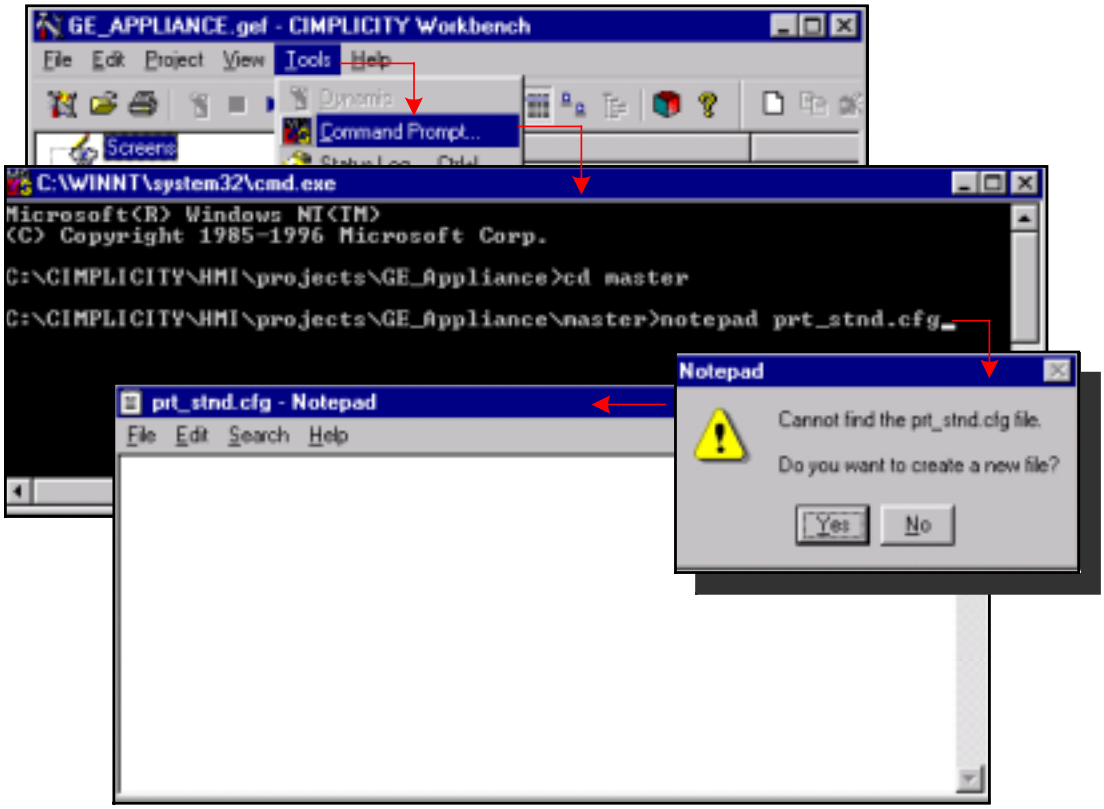
### To configure the prt_stnd.cfg file:

1. Click Tools on the Workbench menu bar.
2. Select Command Prompt.

   A DOS window opens.
3. Enter **cd master** on the Command Line .
4. Press **Enter** to switch to the Master directory.
5. Type **notepad prt_stnd.cfg** (where **Notepad** is the text editor).
6. Press **Enter**

   A message box displays.
7. Click **Yes** in the message box.

***Result: A blank file will display in Notepad called prt_stnd.cfg.***




8. Type the reserved characters, i.e. delimiter, continuation and comment (| - *) on the first line in Notepad.
9. Press **Enter** to go to the next line.

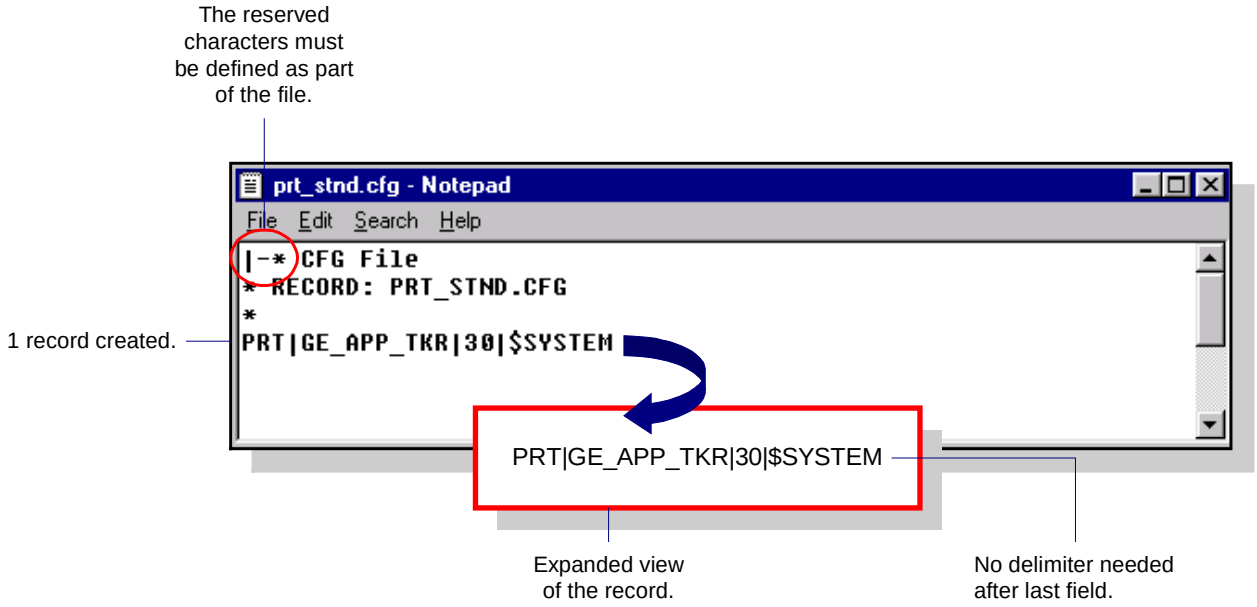
10. Type * RECORD: PRT_STND.CFG.
11. Press **Enter**.
12. Type * and press **Enter**.
13. Create your record using the field descriptions as follows—*see the Configuration File Example*.

| No. | Field Name | Description | Data Type | Entry Example |
|---|---|---|---|---|
| 0 | **server_name** | Base name for the PRT Server that the standby process will support. | 16 characters | PRT |
| 1 | **project_name** | Name of the project that contains the PRT Server. | 32 characters | GE_APP_TKR |
| 2 | **reconnect_time** | The time in seconds between the failure of the PRT Server and the attempt to reconnect. | Integer | 30 |
| 3 | **resource_name** | Resource name that will be used by the standy process to generate alarms. | 16 characters | $SYSTEM |

14. Click File on the menu bar.
15. Select Save.

***Result: The prt_stnd.cfg file is manually configured and saved to the Master directory.***

## *Configuration File Example*



# PRT_WINDOW.CFG

The PRT Window (Field Labels) (**prt_window.cfg**) configuration file is used to customize the columns and their order in the PRT User Interface (PRT_UI). This file has a DEFAULT record already configured. If you want to rearrange or limit the information on screen, you can do so by View or by Region. This feature comes in handy for printing reports and for customizing a screen to display only the information needed by a technician.

Further, this file can be used to configure different views for each of your designated PRT Services, or views to run across all PRT Services.

**OPTIONAL FILE**

**A View record:**

- Is common to all regions.
- Specifies what information displays.
- Arranges information in the display.
- Customizes labels (column headings).
- Uses the keyword DEFAULT in the first line.

**A Region View record:**

- Impacts a specific region.
- Specifies what information displays.
- Arranges information in the display.
- Customizes labels (column headings).
- Overrides other views.
- Uses the keyword REGION in the first line.

**To configure the prt_window.cfg file**:

1. Click Tools on the Workbench menu bar.
2. Select Command Prompt

   A DOS window opens.
3. Type **cd master** on the Command Line .
4. Press **Enter** to switch to the Master directory.
5. Type **notepad prt_window.cfg** (where **Notepad** is the text editor).
6. Press **Enter**.

   The default record for prt_window.cfg will display in the text editor.
7. Do one of the following:

   *Option 1*

   A. To create a common view for all of your regions, type DEFAULT.

   B. Type the delimiter, e.g. |.

   C. Type a name for the view, e.g. VIEW1.

*Option 2*

A. To create a view for a specific region, type REGION.

B. Type the delimiter, e.g. |.

C. Type the name of the region, e.g. ROBOT1.

8. Press **Enter**.

9. Create your record(s) using the field descriptions as follows:

| No. | Field Name | Description | Data Type |
|---|---|---|---|
| 0 | **field_type** | Code defining the field type as follows:<br>0 = Internal<br>1 = Attribute<br>2 = Status | One character |
| 1 | **prt_field** | PRT field for which the modified label will be used. Use the following keywords:<br>**PRT_ASSOC_ITEM**<br>**PRT_EXT_HOLD**<br>**PRT_GROUP_ID**<br>**PRT_INT_HOLD**<br>**PRT_ITEM_ID**<br>**PRT_ITEM_CLASS_ID**<br>**PRT_ITEM_TYPE**<br>**PRT_REFERENCE_ID**<br>**PRT_REGION_ID** | 16 characters |
| 2 | **prt_label** | Label or column head to be alternately used in the PRT window. Recommend title case for alternate labels. | 16 characters |
| 3 | **field_size** | Width of the field by number of characters. | Integer |

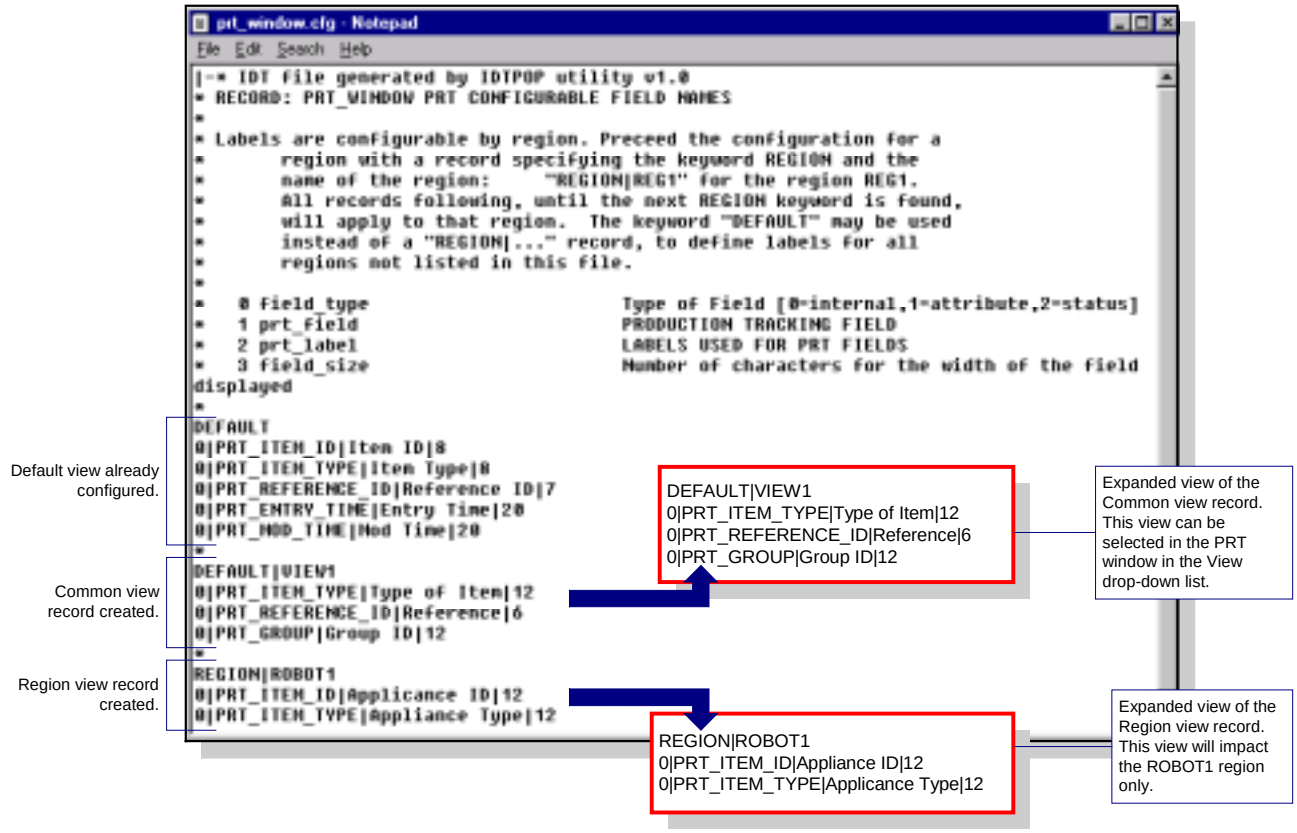
10. Click File on the menu bar after you have created all of your views and proceed as follows:

| Do | When |
|---|---|
| Select Save | The PRT_UI is configured to view all PRT Services. |
| Select Save As | The PRT_UI is configured to view a specific service<br>*Then*<br>A. Enter **aprt_window.cfg** in the File Name field,<br>*Where*<br>**aprt** is the name of the PRT Service<br>B. Click **Save.** |

11. Close the text editor.

***Result: The records created in the prt_window.cfg file will impact the type and arrangement of the information displayed in the PRT window by PRT Service, by common view and by region.***

## Configuration File Example


prt_window.cfg - Notepad
File Edit Search Help
|-* IDT file generated by IDTPOP utility v1.0
* RECORD: PRT_WINDOW PRT CONFIGURABLE FIELD NAMES
*
* Labels are configurable by region. Preceed the configuration for a
*  region with a record specifying the keyword REGION and the
*  name of the region:  "REGION|REG1" for the region REG1.
*  All records following, until the next REGION keyword is found,
*  will apply to that region. The keyword "DEFAULT" may be used
*  instead of a "REGION|..." record, to define labels for all
*  regions not listed in this file.
*
*  0 field_type  Type of Field [0=internal,1=attribute,2=status]
*  1 prt_field  PRODUCTION TRACKING FIELD
*  2 prt_label  LABELS USED FOR PRT FIELDS
*  3 field_size  Number of characters for the width of the field
displayed
*
DEFAULT
0|PRT_ITEM_ID|Item ID|8
0|PRT_ITEM_TYPE|Item Type|8
0|PRT_REFERENCE_ID|Reference ID|7
0|PRT_ENTRY_TIME|Entry Time|20
0|PRT_MOD_TIME|Mod Time|20
*
DEFAULT|VIEW1
0|PRT_ITEM_TYPE|Type of Item|12
0|PRT_REFERENCE_ID|Reference|6
0|PRT_GROUP|Group ID|12
*
REGION|ROBOT1
0|PRT_ITEM_ID|Applicance ID|12
0|PRT_ITEM_TYPE|Appliance Type|12
Default view already configured.
Common view record created.
Region view record created.
DEFAULT|VIEW1
0|PRT_ITEM_TYPE|Type of Item|12
0|PRT_REFERENCE_ID|Reference|6
0|PRT_GROUP|Group ID|12
Expanded view of the Common view record. This view can be selected in the PRT window in the View drop-down list.
REGION|ROBOT1
0|PRT_ITEM_ID|Appliance ID|12
0|PRT_ITEM_TYPE|Applicance Type|12
Expanded view of the Region view record. This view will impact the ROBOT1 region only.

# PRT Client – Routing Items across Project Boundaries

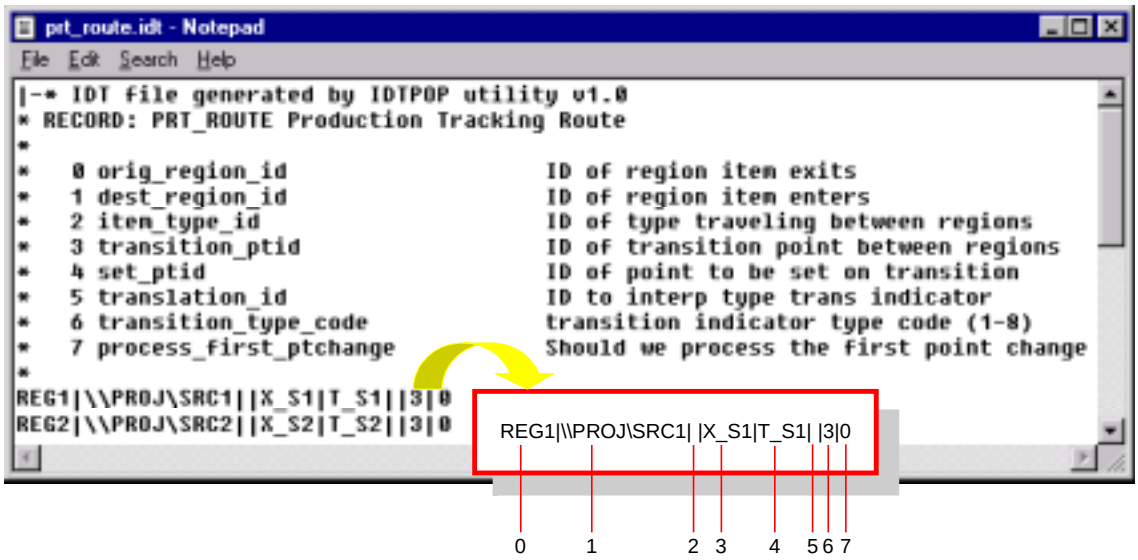
The PRT Client application allows Tracking items to be routed across project boundaries, in the same manner in which items are routed between regions within the same project. In order to facilitate routing across project boundaries, you must configure both the **PRT_ROUTE** and **PRT_INV_TYPE** configuration files. The **dest_region_id** field must fully qualify (spell out) the project and region name, and be less than 16 characters in keeping with the standard for the field. Note that the PRT Client process will ignore the record if the **dest_region_id** field is not fully qualified.

A sample of the configuration needed to run the PRT Client process is detailed using both the **PRT_ROUTE** and **PRT_INV_TYPE** configuration files.

*See "Tracking Route – PRT_ROUTE" and "Tracking Route Invalid Type – PRT_INV_TYPE" sections for further details about the fields in these configuration files.*

## *PRT_Route Configuration for PRT Client*

**PRT_ROUTE Configured to run PRT Client Process**

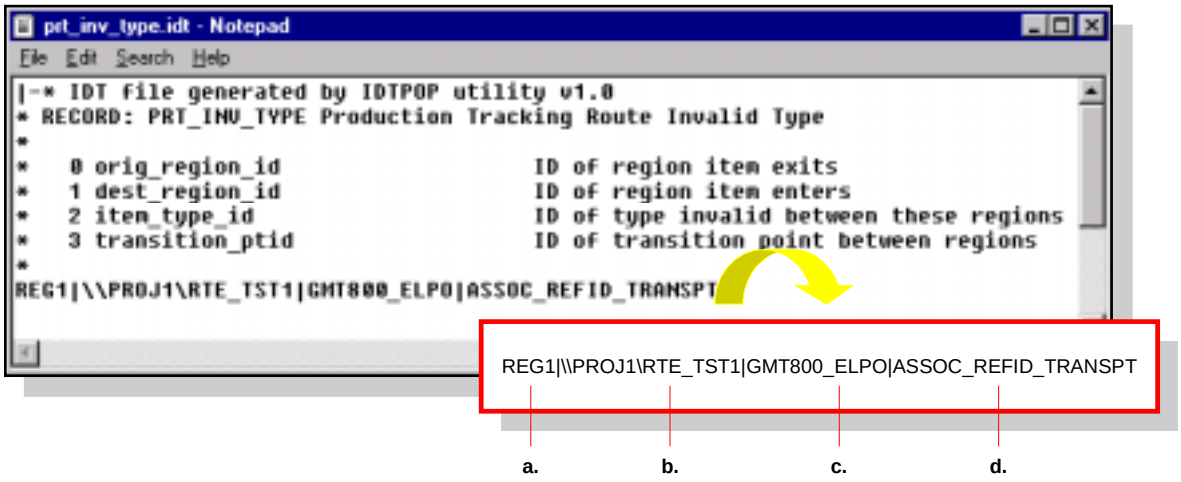

prt_route.idt - Notepad
File Edit Search Help
|-* IDT file generated by IDTPOP utility v1.0
* RECORD: PRT_ROUTE Production Tracking Route
*
* 0 orig_region_id ID of region item exits
* 1 dest_region_id ID of region item enters
* 2 item_type_id ID of type traveling between regions
* 3 transition_ptid ID of transition point between regions
* 4 set_ptid ID of point to be set on transition
* 5 translation_id ID to interp type trans indicator
* 6 transition_type_code transition indicator type code (1-8)
* 7 process_first_ptchange Should we process the first point change
*
REG1|\\PROJ\SRC1||X_S1|T_S1||3|0
REG2|\\PROJ\SRC2||X_S2|T_S2||3|0
REG1|\\PROJ\SRC1| |X_S1|T_S1| |3|0
0 1 2 3 4 5 6 7

### Field Descriptions

0. Source region from which the item will exit.
1. Fully qualified path to the project and destination region to which the item will enter.
2. Item type ID that will be traveling this route; can be left blank if numerous types will be traveling this route.
3. Transition Point ID between the two regions.
4. Point ID to be set when items transition from the source region to the destination region.
5. Translation ID to interpret the type transition indicator; field left blank in example.
6. Transition type code *(see "Tracking Route – PRT_ROUTE" for details).*
7. Process first point change determines whether to process the first point change of a route as a transition point.

## *PRT_Inv_Type Configuration for PRT Client*

**PRT_INV_TYPE Configured to run PRT Client Process**


pet_inv_type.idt - Notepad
File Edit Search Help
|-* IDT file generated by IDTPOP utility v1.0
* RECORD: PRT_INV_TYPE Production Tracking Route Invalid Type
*
*  0 orig_region_id  ID of region item exits
*  1 dest_region_id  ID of region item enters
*  2 item_type_id  ID of type invalid between these regions
*  3 transition_ptid  ID of transition point between regions
*
REG1|\\PROJ1\RTE_TST1|GMT800_ELPO|ASSOC_REFID_TRANSPT
REG1|\\PROJ1\RTE_TST1|GMT800_ELPO|ASSOC_REFID_TRANSPT
a.
b.
c.
d.

### Field Desciptions

A. Source region from which the item will exit.

B. Fully qualified path to the project and destination region to which the item will enter.

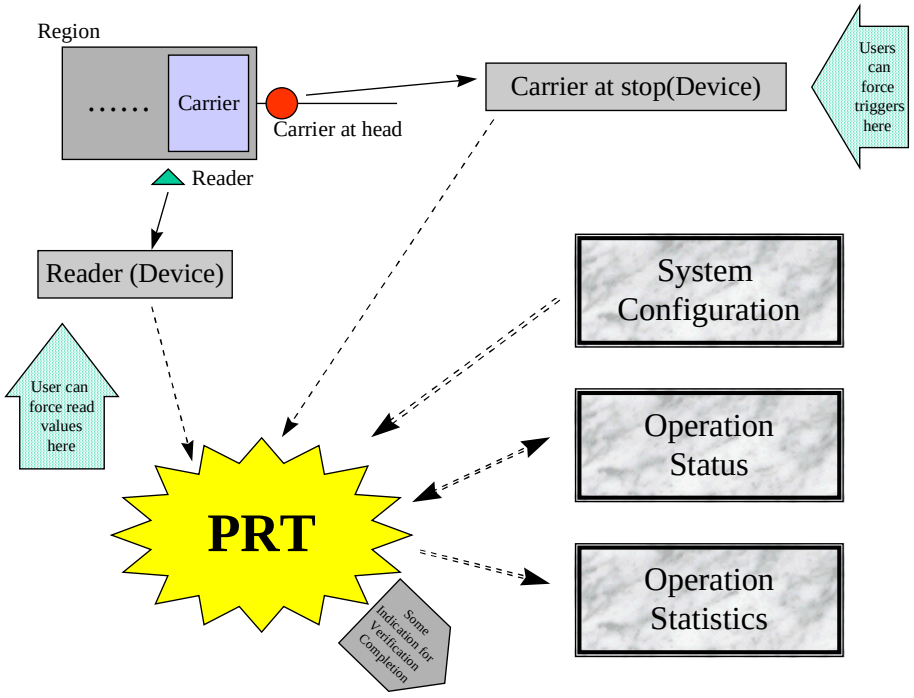
C. Item type ID that is not allowed to travel this route, or the invalid item type ID. Note that if you left this field blank or used the wildcard character in the **PRT_Route** record, you may need more than one record to indentify the invalid item types.

D. Transition Point ID between the two regions.

# Automated Region Validation

Automated PRT Region represents a combination of functions that address the need to effectively process positive identification data (such as, Smarteye, RF Tag or bar code readers). This functionality is used to make sure that the CIMPLICITY HMI Tracking Image is accurate at the head of zones where readers are located, and decisions are not made on an item until it is at the head of a zone and the tracking system has been verified against the reader. This can be used to reduce manual intervention when a read is missed or misread. It also creates an enhanced self-healing capability without the need for single-carrier virtual regions at the head of each region with a reader (this setup was used to utilize the positive ID transitions that made sure that the tail of the destination region was corrected).

The components of the system are:




An overview of the system and its components is shown above. In general terms the following is a list of major components and their roles.

- Reader: Shown with a triangle. The carrier IDs are read by the reader and are put in to a reader point.
- Carrier at head: Shown with a circle. This is a trigger that tells PRT that a carrier has just come to the head of the region, and the reader point should contain the ID of the carrier. This trigger is usually a limit switch at the head of the region indicating that there is a carrier present. Notice that the Reader point should not be used for this trigger if the user wants no-read cases to be detected by this system.
- Manual Entry 1 (Correct Reader Value): If the reader has a no read or the PRT does not match the read, and the region can not be validated. Then the solution should provide a way of manually entering the correct read.

- Manual Entry 2 (Re-triggering the validation process): In case the region validation process fails (this will probably be as a result of bad read or no read), the solution should provide a way of forcing the system to validate itself again. This is usually needed in combination with the Manual Entry 1. The mechanism can either be automatic (via a heartbeat) or manual.

If the users' confidence in readers are high or they are planning a different behavior in case of a no-read, users can set up the system so that reader value can play the role of validation trigger as well. In this case it is users' responsibility make sure that no-read cases are handled.

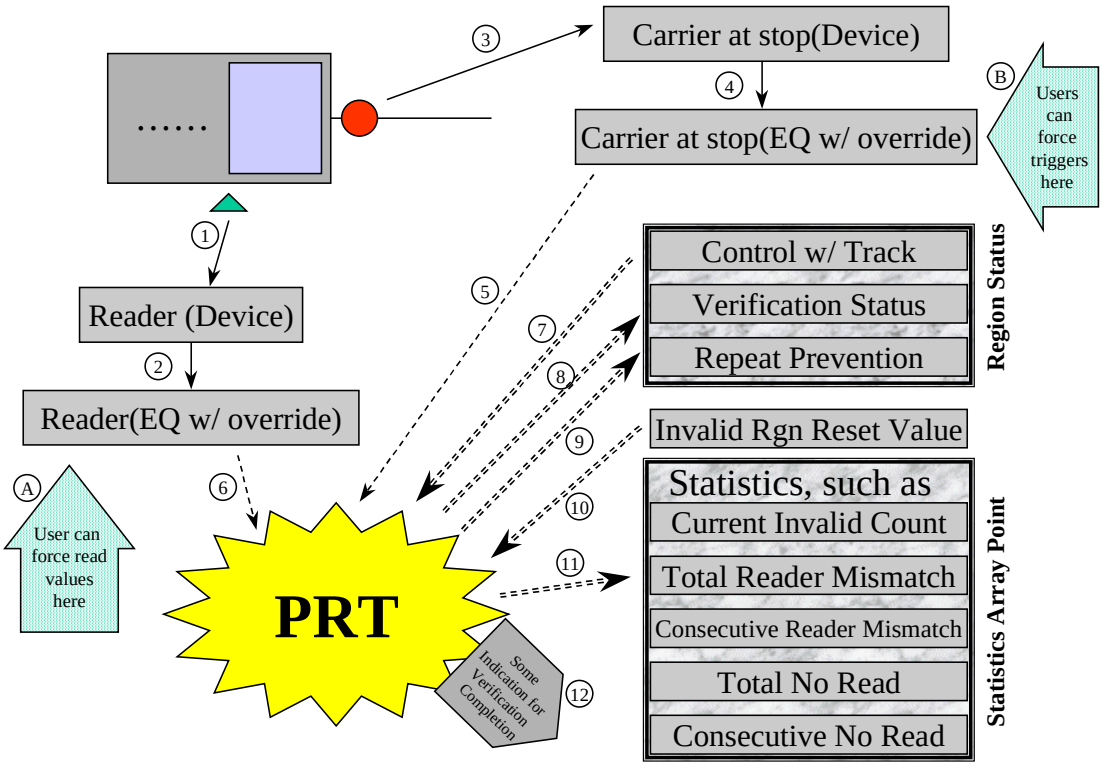
## Sequence of Events for Region Validation

An assumption for this system is that there can only be one carrier between the reader and the head of the region when PRT is triggered for the existence of a carrier at the head of that region. This will eliminate the timing problems that can arise in different circumstances.

Please refer to Figure 2 while the events in this section are described. A carrier passes by the reader at the head of a region. The value of the read (if there is any) is copied to a device point configured for that reader (1). This value is then copied into a derived point of type 'Equation with override' for the reader (2). At this point please note that equation with override derived point types are only available for analog points. This design allows the users to configure analog points for the reader values. These points are then converted to ASCII for internal processing using a region specific 'C' formatting string supplied in the configuration. This derived point will be the mechanism to manually enter the correct reader values if necessary.

After giving the system enough time to get the complete read in to the device point, a trigger lets the PRT subsystem be aware of a new item at the head of the region (3). This is usually a limit switch at the head of the region which acts at 'carrier at head' indicator. This trigger is copied into a derived point similar to the reader point (4). This will provide a mechanism to re-trigger the PRT validation process again.

The components and their relations to each other in the proposed solution to Control with tracking are:




Changes in the derived trigger point will be triggering PRT (5). When that happens, PRT retrieves the latest value of the reader (5) and the configuration for the region on how to proceed with the validation (7). After the validation process is completed PRT updates the results of the process (8)(9) and updates some runtime statistics (11).

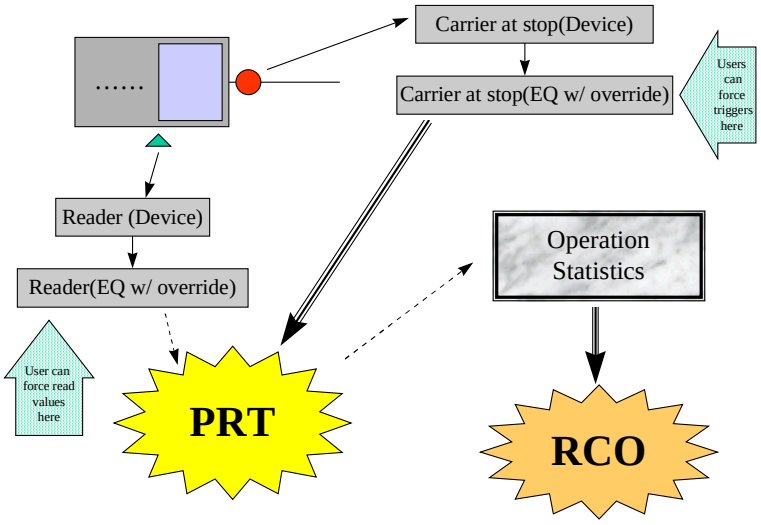
After the validation process is completed, the PRT subsystem should provide a mechanism for letting other subsystems know about the completion of the validation process (12). This would help eliminate the timing issues that might arise between the PRT validation process and the processes (such as RCO s or an event manager driven routing mechanism) that may act on the validity of regions after the validation process has completed. Since validation statistics are going to be updated after every validation, this point will be used to trigger other processes.

## Reviewing Different Scenarios for Flow of Control

In this section different plant setup scenarios that will result in different operational behavior are illustrated.

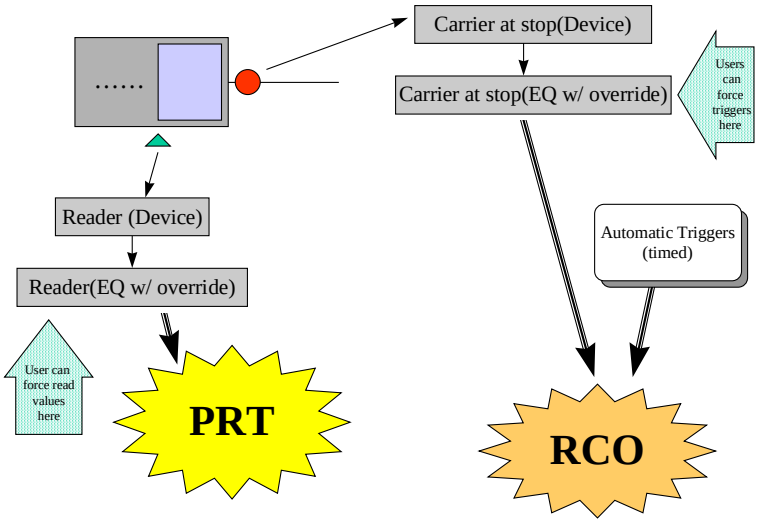
### *Setup 1–Flow of Control*

PRT validation is triggered by the ‘carrier at stop’ point. PRT does its validation and then sets the statistics point, which in turn triggers the RCO.



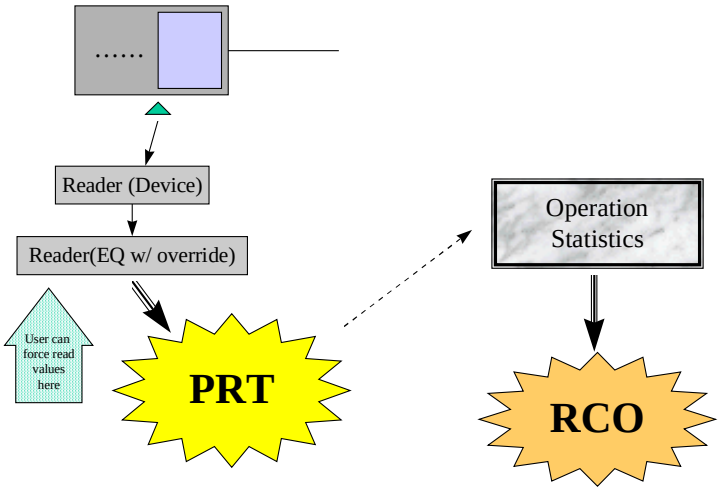
### Setup 2–Flow of Control

PRT validation is triggered with the reader point. In this case, PRT can not catch no-read cases. The systems relies on the processing performed by the RCO when the carrier sets the carrier at stop point. In the case of manual intervention (setting the correct reader value), the RCO can be triggered by the user or the RCO can be configured to re-trigger automatically.



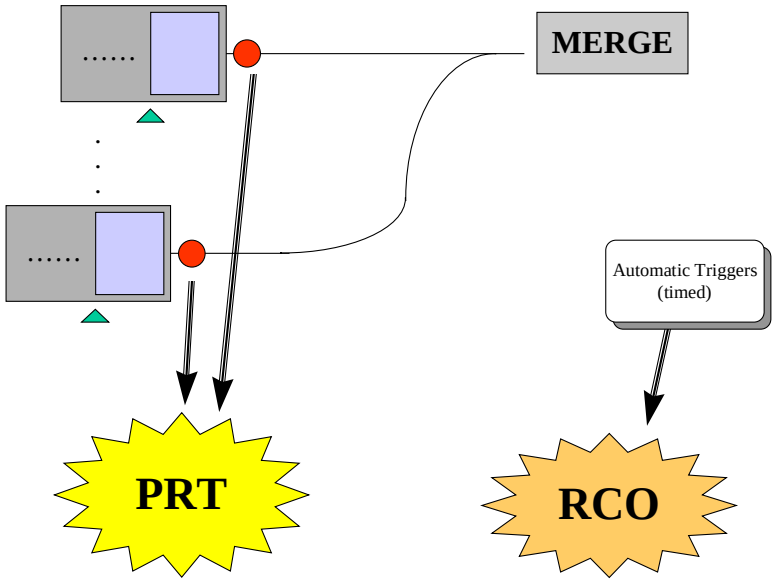

### Setup 3–Flow of Control

In this configuration, there is no carrier at stop point. The system relies on the readers being correct most of the time.



### *Setup 4–Flow of Control*

This is a combination of Configuration 1 and 2. Handling of the No-Read cases for the regions are handled in PRT. The RCO is then triggered independently with a heartbeat. This automatic trigger can be combined with the validation triggers. Note that because of multiple regions, some of the items shown in the previous configurations are eliminated to make the figure more readable.




## Accessing Configuration files by the Subsystem

This section describes any files accessed by the subsystem, and provides specifics about how this data is used.

### *prt_reg_val.cfg*

The following table gives the layout this text file. Region specific configuration is done with this file.

| | Description |
|---|---|
| **Region_id** | Region id |
| **Item_or_ref** | Type of matching to perform. Valid values are:<br>1 = match item_id<br>2 = match reference_id |
| **Reader_point_id** | Reader values are provided with this point. It can be a string or numeric point. The reason that we are providing a numeric point as reader point is that Equation with override derived point is only applicable to numeric points. |
| **Class_id** | Class Id to use for matching (used if multiple item per location configured, and if only one item type is being identified. For example, PRT can be configured to contain 2 items per location, one carrier and one buck, and the readers in the system read the id's of carriers. In this case the user configures PRT so that carrier item types and buck item types are in different classes; and uses carrier class in this field) |

| | |
|---|---|
| **Validation_trigger** | When received, depending on the next field, used to trigger the validation process. This field is ignored if the next field is Use_Reader. |
| **Validation_trigger_type** | Validation trigger types are:<br>1 = Transition_high<br>2 = Transition_low<br>3 = Limit_switch<br>4 = Use_Reader_point |
| **Statistics_point_id** | Point id to put the run time statistics. It should be a DWORD array of 9 elements, but if it is less than 9 only a portion of the statistics will be written. |
| **Reader_format** | It is a C print formatting string. It should be in quotes. For string points use variations of %s, for analog points use variation of %ld. If empty "%s" (or "%ld" depending on the point type) is used. Reader values are formatted with this string before trying to match it to the head item's item_id or reference_id. |
| **Invalid_reset_value** | When a region has a mismatch with reader value, region image is assumed invalid. Region's invalid count is set to this value. This count shows the number of consecutive good reads that is necessary to make the region valid again. |
| **Control_With_Track** | This value is used when backing files are created for the first time. |
| **Use_OOS** | If the region image does not match the read, and the item that should have been at the head of the region is not at the head, but it is in the same region; Automated Region Validation will move the items that are in front of the identified item to the detainment region of the region. The move is done as item moves if this flag is set to 0. Otherwise (if it is 1), out of sequence processing feature of PRT will be used to determine how those items will be treated. |

## prt_svc_val.cfg

The following table gives the layout of this text file. Service specific configuration is done with this file. Command_point and response_point will be used to expand the functionality of PRT without changing the interface to it.

Command point has the following general format:

**<***REFERENCE***>,<***COMMAND***>,<***PARAMETER-LIST***>**

*Where*

**<***REFERENCE***>** is copied back to the response point, and can be used by the clients to match the responses to the command that they have sent. The following is the list of commands supported and their formats:

**<REFERENCE>,$RESET_STATS,{<REGION_ID>|$ALL},<INDEX>**

**<REFERENCE>,$RESET_PREV_RD,{<REGION_ID>|$ALL}**

Note that $ALL is used to refer to the all regions of the PRT service. If **<***INDEX***>** is 0, all the elements of the statistics point will be set to 0. Otherwise, the specified element will be set to 0.

The field descriptions are:

| Field | Description |
|---|---|
| **Service_id** | Service id |
| **Warning_alarm_id** | Alarm Id to use for validation alarms that PRT was able to fix. If not configured, the alarms are logged to cor_status.log. |
| **Failure_alarm_id** | Alarm Id to use for validation alarms that PRT was not able to fix, and requires manual correction of the system. If not configured, the alarms are logged to cor_status.log. |
| **Command_point** | PRT command point |
| **Response_point** | PRT response point |
| **keep_validation_flags** | If it is 1, it means to keep the validity flags from the global memory when PRT starts up. If set to 0, reset the validity flags to FALSE regardless of their values in the global memory. |

## Runtime Data

The validation process is also required to maintain some statistics about the performance. The following table gives the list of the elements of the statistics point.

| Index | Field | Description |
|---|---|---|
| 0 | MODIFY_MASK | Collection of bits giving information about the validation process just completed. Details of the bits are given in the following table. |
| 1 | TOTAL_VALIDATIONS | Number of times the region validation is triggered. |
| 2 | INVALID_COUNT | Current ‘Region Invalid’ count. Region is Invalid if this count is greater than 0. |
| 3 | TOTAL_INVALID | Number of times the region has become invalid |
| 4 | TOTAL_DBL_TRIGGER | Number of times the region is double triggered (Triggered multiple times, with the same read while the head is validated) |
| 5 | TOTAL_MISMATCH | Number of times head of the region did not match the reader value |
| 6 | CONS_MISMATCH | Consecutive mismatch count |
| 7 | TOTAL_NOREAD | Number of times region validation is triggered with no new read. |
| 8 | CONS_NOREAD | Consecutive no read count |

### *MODIFY_MASK Bits*

The first element of the statistics point contains information about the validation process just performed. This information is in the form of bits that are described below.

| Bit | Field | Description |
|---|---|---|
| 0 | HEAD_ITEM_VALID | The Head Item of the region was valid when the point was set. |
| 1 | TOTAL_VALIDATIONS_MOD | Statistics array point's value at index 1 has been modified |
| 2 | INVALID_COUNT_MOD | Statistics array point's value at index 2 has been modified |
| 3 | TOTAL_BECOMING_INVALID_MOD | Statistics array point's value at index 3 has been modified |
| 4 | TOTAL_DOUBLE_TRIGGERS_MOD | Statistics array point's value at index 4 has been modified |
| 5 | TOTAL_MISMATCH_MOD | Statistics array point's value at index 5 has been modified |
| 6 | CONSECUTIVE_MISMATCH_MOD | Statistics array point's value at index 6 has been modified |
| 7 | TOTAL_NOREAD_MOD | Statistics array point's value at index 7 has been modified |
| 8 | CONSECUTIVE_NOREAD_MOD | Statistics array point's value at index 8 has been modified |
| 27 | RESERVED_READ | Read is ignored (reserved read starting with a $). |
| 28 | STATS_RESET | Statistics point has been reset |
| 29 | NEW_ITEM | Item not found in the current PRT Service, and a new item has been created |
| 30 | REORDER | Region has been validated by reordering the region. |
| 31 | MOVE | Region has been validated by moving an item from another region. |

## PRT Region Status Bits

Region status bits include:

- Control with Track Flag
- Validation Status
- Waiting For Decision Flag
- Deriving Region Status from Validation Process Data
- PRT Validation completed trigger.
- Reader Value
- PRT Validation trigger.

### *Control with Track Flag*

If set to TRUE, when there is a no read, it tells the validation process to trust the PRT image of the region if the region is valid. This is useful in cases such as region validity is not being mission critical, and having a bad reader.

The Control with Track flag is initialized with the value in prt_reg_val.cfg when the tracking image is created the first time (when they do not exist). It can be changed dynamically with an API call. Since it will be maintained in the region status bits, its value will be maintained when PRT is restarted and when PRT import / export utilities are run.

### *Validation Status*

When an item comes to the head of a region, PRT is triggered to validate the item. If the PRT image matches the reader value, then this flag is set to TRUE, other wise it is reset to FALSE. PRT will reset this flag to FALSE when the head item of the region leaves its location.

### *Waiting For Decision Flag*

After a routing controlling process (such as RCO) makes a decision on a region, it should not make another decision until the current one is executed. This may pose a problem in certain routing configurations such as a merge. Since merges will have multiple source regions, RCO sites will most likely have multiple triggers. This will become a problem if two triggers (especially heartbeat triggers) result in the same decision before the previous one is executed. To resolve this repeated decision problem, it is proposed to introduce a Waiting For Decision flag that is associated with the region. The routing controlling process sets this flag when it makes a decision involving release of an item from that region. It is reset when a new item validation trigger is received.

## Deriving Region Status from Validation Process Data

The following table lists the region status determination based on the region status variables.

| Region Status | Valid Flag | Waiting for Decision |
|---|---|---|
| Startup Condition (A){Unknown/Undefined} | × | × |
| New Item at head and Validated (B) | ✓ | ✓ |
| New Item at head and not Validated (C) | × | ✓ |
| Item at head Validated and used (D) | ✓ | × |

To help understand the values of these status flags based on the events, the following state diagram is provided.

## PRT Validation Completed Trigger

In order to avoid timing issues, the routing process and the PRT validation should not both be triggered by the same trigger. Since the validation statistics will be updated after validation is completed, this point can be used to trigger the routing process. This will help minimize the configuration requirements as well.

## Reader Value

This is how PRT will retrieve the latest read from the reader at the head of the region. Since the PRT item id (or reference id) is a string of 16 characters, it is recommended to configure at least a string point. Even though there are not any size constraints, it is suggested to use a string type with less than 16 characters. If the region validation is configured with items containing a region specific prefix, this prefix is added to the front of the reader value before it is used to match head item's item id (reference id).

If users will be allowed to correct reader values in case of a no read or bad read, it is suggested to use an 'equation with override' derived point which will be fed from the real reader point.

## PRT Validation Trigger

This point will be used to trigger PRT to validate the new head item in the region. Even though using Reader Value point for this purpose may seem logical, this would eliminate the PRT Validation process's capability to determine no read conditions. Users can still configure reader value points as PRT validation trigger points.

If users will be allowed to force validation, it is suggested to use an 'equation with override' derived point which will be fed from the real trigger point.

# Configuration Validation Program

The Configuration Validation Program can be run to produce a report if errors are found in the configuration. In order to run the program:

Open a DOS window and change directory to the data directory for your project and then execute the program by typing the following command at the prompt:

**PRTV <***filename***>**

*Where*

**<***filename***>** is the error report file name.

You will receive one of the following two messages:

- No errors found
- Errors found - report in file **<***filename***>**.

Sample reports are found below (an asterisk in the error line means that the error is fatal):

```
                                                        Jan 22 1991 14:41:51


                          PRT_SERVICE Error Report
                  Production Tracking Service Identification File
                  -----------------------------------------------

PRT_SVC_ID Value   | Error found
-------------------|-----------------------------------------------------------
PRT3_DC            | xlat - no known translation srv>ext_hold_svc_id node> obj>
PRT3_DC            | xlat - no known translation srv>ext_item_svc_id node> obj>
PRT3_DC            | xlat - no known translation srv>PRT3_DC node> obj>

                                                        Jan 22 1991 14:41:51

                          PRT_GROUP Error Report
                       Production Tracking Group File
                       ------------------------------

GROUP_ID Value  | Error found
----------------|------------------------------------------------------------
GROUP X         | Specified Resource does not exist  XX_FR

                                                        Jan 22 1991 14:41:51

                          PRT_REGION Error Report
                       Production Tracking Region File
                       ------------------------------

REGION_ID Value | Error found
----------------|------------------------------------------------------------
TEST_REGION     | POINT not configured: PTP06_Y
TEST_REGION     | POINT not configured: PTP06_X
TEST_REGION     | Specified Resource does not exist  FRX
TEST_REGION     |*SERVICE not configured: PRT4_DC
```

# Production Tracking Import Program (PRT_IMPORT)

This section provides a description of the *Production Tracking Import Program* (**PRT_IMPORT**). It describes the tasks performed by the Import Program and defines Import file record formats (for each type of command the Import Program can send to PRT).

A set of Import record types and their corresponding record formats are defined. Each of these record types is associated with a particular PRT function. The remainder of the section describes the Import file record formats for each record type, and how to invoke the program.

## Program Elements (Production Tracking Import Program)

The Production Tracking Import Program interprets a file to determine the commands to issue. It also provides information about the results of the commands issued.

- Process Registration
- Initialization of the Production Tracking Application Interface
- Determination of the name of the Import data file to be used. The name of the input file is specified on the command line, (for example, **prt_import <filename>**). The input file is assumed to have **.prtm** as a filename extension. If a filename is not specified on the command line, the program prompts for it.
- Opening of the Import data file
- Reading of Import file records
- Parsing of Import file records to determine applicable PRT Application Interface routine to call
- Further parsing of Import file record to obtain the arguments to be included to the PRT Application Interface call
- Call to appropriate PRT Application Interface routine
- Reporting of any errors returned by the Application Interface.
- Termination of the Application Interface
- Reporting of "command" totals for each type of command, total records processed, etc.

## Import File Naming Conventions

The file naming conventions are:

| | |
|---|---|
| Input Files: | <*filename*>**.PRTX** |
| Output Files: | <*filename*>**.PRTM** |

*Where*

**<***filename***>** for input is used to create output.

## Import File Format

The Import program reads a series of records from the Import file. Each record begins with a 3-character command. Valid commands are as follow:

| | |
|---|---|
| ADD | Add item to be tracked |
| ADV | Advance item along specified route |
| ATT | Add item attribute |
| CLL | Clear location |
| DEL | Delete tracked item |
| DSP | Print message to terminal |
| HLD | Apply group hold to item |
| INS | Insert item to be tracked |
| MAT | Modify item attribute |
| MOD | Modify tracked item |
| MOV | Move item from source region to destination region |
| PTS | Enable/disable (1/0) production setpoints |
| REG | Modify region status |
| STT | Initiate a production start |
| STP | Initiate a production stop |

Subsequent fields in the record are delimited by a field delimiter character (usually |) defined on the very first line of the Import file.

**Note:** A blank line in the import input file causes the Import program to terminate processing of the input file at that point.

All fields in the file are of type character. The actual data passed to PRT is converted to the correct type by the Import program.

- For the DELete command the disposition can be specified as "SCRAP", "DEL", or "DELETE". These strings are translated to the manifest constants SCRAP and DELETE.
- For any API functions that require a *region location* argument (e.g. the ADD and INSert commands) the region location can be specified as "ALL", "LAST" ("ENTRY"), or "FIRST" ("EXIT"). These are translated to the manifest constants ALL_LOCATIONS, LAST and FIRST, respectively.
- For arguments that are required to be a Boolean value, the strings "T", "TRUE", "F", or "FALSE" can be specified, which translate to the manifest constants TRUE and FALSE.

## ADD (Add an Item to be Tracked)

This command provides a mechanism to add an individual Item to a location in which another Item currently resides.

The fields in this record are:

| | |
|---|---|
| `char record_type[3]` | ADD |
| `COR_BOOLEAN ext_proc_flag` | If TRUE, get all info from ext process for serialized item |
| `char region_id[17]` | Region where the item is to be added to the queue or production start is to begin |
| `COR_I2 region_loc` | Specific location in the region where the Item is to be added |
| `char item_id[21]` | Unique identifier of a serialized item |
| `char reference_id[21]` | Secondary identifier of a serialized item |
| `char item_type_id[17]` | Item type identifier |
| `char parent_item_id[21]` | Associated serialized item identifier |
| `COR_U4 item_status` | Item status |
| `COR_BOOLEAN ext_hold_active` | If TRUE, item held due to external hold. |
| `char group_id[17]` | Identifier of the Tracking Group providing the reason for the hold. |
| `COR_BOOLEAN int_hold_active` | If TRUE, item held due to internal hold. |
| `char int_hold_reason[41]` | Comment specifying reason for internal hold. |
| `char comment[41]` | Comment to be recorded in prt history log file. |
| `char user_or_svc_id[33]` | User or service identifier |

If no Item Attribute information is to be specified for this item, the ADD record need not be preceded by any ATT records (for filling in elements within the Item Attribute array). To specify Item Attribute information, see the description of the ATT record type.

## ADV (Advance an Item Along a Specified Route)

The Application Interface call for this record type provides a mechanism for advancing the next Item Carrier to leave a Source Tracking Region to a Destination Tracking Region. This call moves ALL items at the FIRST (EXIT) location in the source region to the LAST (ENTRY) location in the destination region.

The fields in this record are:

| | |
|---|---|
| `char record_type[3]` | ADV |
| `char src_region_id[17]` | Region the item is moving out of |
| `char dest_region_id[17]` | Region the Item is moving into |
| `char comment[41]` | Comment to be recorded in prt history log file |
| `char user_or_svc_id[33]` | User or service identifier |

## ATT (Add an Item Attribute)

This record type provides a means of filling in the array of Item Attribute structures (one element at a time) the program maintains internally, for providing as an argument to the PRT Application Interface functions invoked by the STT, ADD, INS, and MAT record types.

The Item Attribute structure is:

```
typedef struct
{
  char att_name[17];
  char att_value[17];
} PRT_ITEM_ATT;
```

The fields in this record are:

| | |
|---|---|
| `char record_type[3]` | ATT |
| `char att_name[17]` | Attribute name |
| `char att_value[17]` | Attribute value |

The array of **`PRT_ITEM_ATT`** structures has **`PRT_NUM_ATTRIBUTES`** elements. This is the maximum number of ATT records that can appear in the Import file without an intervening **`STT, ADD, INS, or MAT`** record (which sends whatever Item Attribute data is currently in the array and then clears it out for the next Import command (API call) that uses it). More than **`PRT_NUM_ATTRIBUTES ATT`** records without any intervening STT, ADD, INS, or MAT records will result in an error being reported to the user.

## CLL (Clear Location)

The Application Interface call for this record type provides a mechanism for deleting or scrapping all items at a given location in a specific Tracking Region.

The fields in this record are:

| | |
|---|---|
| `char record_type[3]` | CLL |
| `char region_id[17]` | Identifier of the Tracking Region where the Item is to be deleted from the queue |
| `COR_I1 disposition` | Manifest constant specifying the disposition of the item (which can be specified as 'DEL', 'DELETE' or 'SCRAP'.). |
| `char comment[41]` | Comment to be recorded in prt history log file |
| `char user_or_svc_id[33]` | User or service identifier |

### DEL (Delete a Tracked Item)

The Application Interface call for this record type provides a mechanism for deleting or scrapping a single Item from all configured PRT services, a specific PRT service, a specific Tracking Group or a specific Tracking Region.

The fields in this record are:

| | |
|---|---|
| `char record_type[3]` | DEL |
| `char prt_svc_id[33]` | PRT service identifier |
| `char group_id[17]` | Identifier of the Tracking Group |
| `char region_id[17]` | Identifier of the Tracking Region where the Item is to be deleted from the queue |
| `COR_I2 region_loc` | Region location where the item resides |
| `char item_id[21]` | Unique identifier of a serialized item |
| `reference_id[21]` | Secondary identifier of a serialized item |
| `char item_type_id[17]` | Item type identifier |
| `COR_I1 disposition` | Manifest constant specifying the disposition of the item (which can be specified as 'DEL', 'DELETE' or 'SCRAP'.). |
| `char comment[41]` | Comment to be recorded in prt history log file |
| `char user_or_svc_id[33]` | User or service identifier |

### DSP (Print Message to Terminal)

This record type provides a means of printing a message out on the terminal. Messages can be inserted in the Import input file to notify the user as to the current point of processing within the file.

The fields in this record are:

| | |
|---|---|
| `char record_type[3]` | DSP |
| `char display_string` | Message to be displayed on terminal. The message can be as long as necessary. |

### HLD (Apply a Group Hold to an Item)

The application interface call for this record type provides a mechanism to activate and deactivate a HOLD specification for a Tracking Group.

The fields in this record are:

| | |
|---|---|
| `char record_type[3]` | HLD |
| `char group_id[17]` | Unique identifier of tracking group |
| `char ext_hold_reason[41]` | Text describing the reason for placing holds on Tracking Regions in this Tracking Group (optional for activate, does not apply to deactivate) |
| `COR_BOOLEAN act_hold_flag` | If set to TRUE, the Hold Specification is activated. If set to FALSE, the Hold Specification is Deactivated |
| `char comment[41]` | Comment to be recorded in prt history log file |
| `char user_or_svc_id[33]` | User or service identifier |

## *INS (Insert an Item to be Tracked)*

This command provides a mechanism for inserting an Item into the Tracking Region queue. If an Item is to be inserted into the queue at a particular location (other than at the head or tail of the queue), the region location of the Item it is to be inserted ahead of is specified (i.e. specifying 'region_loc = 1', is equivalent to specifying 'region_loc = FIRST'). Defined constants are provided to indicate that the item is to be added either at the head or at the tail of the queue. The region location can be specified as FIRST (or also as EXIT) to insert the Item as the first (head) Item in the queue (i.e. this Item will be the next Item to transition out of the queue). LAST and ENTRY are provided to insert the Item as the last Item at the tail of the queue, BEHIND all other Items.

The fields in this record are:

| | |
|---|---|
| `char record_type[3]` | INS |
| `COR_BOOLEAN ext_proc_flag` | If TRUE, get all info from ext process for serialized item |
| `char region_id[17]` | Region where the item is to be added to the queue or production start is to begin |
| `COR_I2 region_loc` | Location of the Item that the inserted item is to be inserted in front of (nearer to exiting the queue). |
| `char item_id[21]` | Unique identifier of a serialized item |
| `char reference_id[21]` | secondary identifier of a serialized item |
| `char item_type_id[17]` | Item type identifier |
| `char parent_item_id[21]` | associated serialized item identifier |
| `COR_U4 item_status` | Item status |
| `COR_BOOLEAN ext_hold_active` | If TRUE, item held due to external hold. |
| `char group_id[17]` | Identifier of the Tracking Group providing the reason for the hold. |
| `COR_BOOLEAN int_hold_active` | If TRUE, item held due internal hold. |
| `char int_hold_reason[41]` | Comment specifying reason for internal hold. |
| `char comment[41]` | Comment to be recorded in prt history log file. |
| `char user_or_svc_id[33]` | User or service identifier |

If no Item Attribute information is to be specified for this Item, the INS record need not be preceded by any ATT records (for filling in elements within the Item Attribute array). To specify Item Attribute information, see the description of the ATT record type.

## *MAT (Modify an Item's Attributes)*

The application interface call for this record type provides a mechanism for modifying Serialized Item Attribute data.

The fields in this record are:

| | |
|---|---|
| `char record_type[3]` | MAT |
| `char region_id[17]` | Unique identifier of the tracking region where the item resides |
| `char item_id` | Unique identifier of a Serialized Item |
| `char reference_id[21]` | Secondary identifier of a Serialized Item |
| `char comment[41]` | Comment to be recorded in prt history log file. |
| `char user_or_svc_id[33]` | User or service identifier |

The purpose of this call is to modify an item's attributes, the MAT record must be preceded by one or more ATT records (for filling in elements within the Item Attribute array). To specify item attribute information, see the description of the ATT record type.

## *MOD (Modify a Tracked Item)*

This command provides a mechanism to do the following:

- Modify Item Tracking Data;
- Associate an Item Type with an unknown Item;
- Associate a Reference ID with an Item ID;
- Associate two Items.

An unknown Item is assigned an Item Type by modifying the Item Type field to the correct value. A Reference ID is associated with an Item ID by modifying the Reference ID for the Serialized Item. A Non-serialized Item is associated with a Serialized Item by modifying the Non-serialized Item to include the new or modified Serialized parent Item ID.

The fields in this record are:

| Field | Description |
|---|---|
| char record_type[3] | MOD |
| COR_U4 modify_bitmask | Indicates field that has been modified<br>These are codes that can be used to indicate which bits in the modify_bitmask are set:<br>I = SET_ITEM_ID_MOD_BIT( result )<br>R = SET_REFERENCE_ID_MOD_BIT( result )<br>T = SET_ITEM_TYPE_BIT( result );<br>P = SET_PARENT_ITEM_BIT( result )<br>S = SET_ITEM_STATUS_BIT( result )<br>H = SET_INT_HOLD_BIT( result )<br>E = SET_EXT_HOLD_BIT( result ) |
| COR_U4 status_bitmask | Indicates fields in item status which have been modified |
| char region_id[17] | Region where item last resided |
| COR_I2 region_loc | Specific location in the region where the no-serialized item resides |

| | |
|---|---|
| char item_id | Unique identifier of a Serialized Item |
| char reference_id[21] | Secondary identifier of a Serialized Item |
| char item_type_id[17] | Item type identifier |
| char parent_item_id[21] | Pertains only to Non-serialized Items |
| COR_U4 item_status | Item status |
| COR_BOOLEAN ext_hold_active | If TRUE, item held due to external hold |
| char group_id[17] | Identifier of the Tracking Group providing reason for hold |
| COR_BOOLEAN int_hold_active | If TRUE, item held due to internal hold |
| char int_hold_reason[41] | Comment specifying reason for internal hold |
| char comment[41] | Comment to be recorded in prt history log file |
| char user_or_svc_id[33] | User or service identifier |

### *MOV (Move an Item from a Source to a Destination Region)*

The Application Interface call for this record type provides a mechanism to move an Item from the Tracking Region where it currently resides to a new Tracking Region. It also permits the movement of an individual Item from one location to another within the same Tracking Region.

The fields in this record are:

| | |
|---|---|
| `char record_type[3]` | MOV |
| `char src_region_id[17]` | Region the item is moving out of |
| `char dest_region_id[17]` | Region the Item is moving into |
| `COR_I2 src_region_loc` | Sequence number of the item in the source tracking region |
| `COR_I2 dest_region_loc` | Sequence number of the item in the destination tracking region |
| `COR_BOOLEAN dest_insert_flag` | If TRUE, the moved Item is inserted in the destination Tracking Region queue ahead of the Item located at the specified destination region location. |
| `char item_id[21]` | Unique identifier of a Serialized Item |
| `char reference_id[21]` | secondary identifier of a Serialized Item |
| `char item_type_id[17]` | Item type identifier |
| `char comment[41]` | Comment to be recorded in prt history log file |
| `char user_or_svc_id[33]` | User or service identifier |

### PTS (Enable/Disable Region Setpoints)

The Application Interface call for this record type provides a mechanism to enable and disable the Item transition setpoints configured in the PRT Tracking Route records. These points are set by PRT when an Item transitions into the configured destination region.

The fields in this record are:

| | |
|---|---|
| `char record_type[3]` | PTS |
| `char region_id[17]` | Tracking region identifier |
| `COR_BOOLEAN enable_point_flag` | If TRUE, enable region item transition setpoint. If FALSE, disable region item transition setpoint. |
| `char comment[41]` | Comment to be recorded in prt history log file |
| `char user_or_svc_id[33]` | User or service identifier |

### REG (Modify Region Status)

The application interface call for this record type provides a mechanism for modifying Tracking Region Data; specifically, the region status.

The fields in this record are:

| | |
|---|---|
| `char record_type[3]` | REG |
| `COR_U4 status_bitmask` | Indicates field that has been modified |
| `char region_id[17]` | Unique identifier of the tracking region |
| `COR_U4 region_status` | Tracking Region status code (e.g. IN-LOCKED, OUT-LOCKED, FULL, OUT-OF-SEQ) |
| `char comment[41]` | Comment to be recorded in prt history log file |
| `char user_or_svc_id[33]` | User or service identifier |

## *STT (Initiate a Production Start)*

Provides a mechanism to initiate a production start for an item in a tracking region.

The fields in this record are:

| | |
|---|---|
| `char record_type[3]` | STT |
| `BOOLEAN ext_proc_flag` | If TRUE, get all info from extprocess for serialized item |
| `char region_id[17]` | Region where the item is to be added to the queue or production start is to begin |
| `char item_id[21]` | Unique identifier of a serialized item |
| `char reference_id[21]` | Secondary identifier of a serialized item |
| `char  item_type_id[17]` | Item type identifier |
| `char parent_item_id[21]` | Associated serialized item identifier |
| `COR_U4 item_status` | Hexadecimal representation of desired item status |
| `COR_BOOLEAN ext_hold_active` | If TRUE, item held due to external hold. |
| `char group_id[17]` | identifier of the Tracking Group providing the reason for the hold. |
| `COR_BOOLEAN int_hold_active` | If TRUE, item held due to internal hold. |
| `char int_hold_reason[41]` | Comment specifying reason for internal hold. |
| `char comment[41]` | Comment to be recorded in prt history log file. |
| `char  user_or_svc_id[33]` | User or service identifier |

If no Item Attribute information is to be specified in the production start for an item, the STT record need not be preceded by any ATT records (for filling in elements within the Item Attribute array). To specify item attribute information, see the description of the ATT record type.

## *STP (Initiate a Production Stop)*

The application interface call for this record type provides a mechanism for removing an Item from the Production Tracking system as the result of its exiting out of a region along a route that has no destination region. The specified Item must currently reside at a source region associated with such a route. If the Item cannot legitimately transition out of the system along such a route, an error is returned.

The fields in this record are:

| | |
|---|---|
| `char record_type[3]` | STP |
| `char region_id[17]` | Identifier of the Tracking Region where the Item to exit the system is located |
| `COR_I2 region_loc` | Specific location in the region where the Item is to be added |
| `char item_id[21]` | Unique identifier of a serialized item |
| `char reference_id[21]` | Secondary identifier of a serialized item |
| `char item_type_id[17]` | item type identifier |
| `char comment[41]` | Comment to be recorded in prt history log file |
| `char user_or_svc_id[33]` | User or service identifier |

## Running the Import Program

To run the Import program, first open a command prompt window. This can be done from the program manager window or a better alternative is from your CIMPLICITY HMI project window. This is better because your path to **%BSM_ROOT%** will be set automatically for you. Change directories to where the import file resides.

Then enter the following command to run the program:

**$ PRT_IMPORT <***filename***>.PRTM <prt_system> <***prt_node***>**

*Where*

**<***filename***>.**PRTM is the import file

**<***prt_system***>** is the prefix of the PRT data server you wish to import to. Specify ALL if all data servers should be connected.

*<prt_node>* is the node where the PRT servers are running. Specify "LOCAL" if they are on the same node.

If you don't specify any of the parameters on the command line, you will be prompted.

**Example Import File**

```
|-*
*
*   Region: R1_BOX
*
REG|FFFF|R1_BOX|0|PRT_EXPORT|
INS|F|R1_BOX|1|||REARBOX||38|F||F||PRT_EXPORT|
INS|F|R1_BOX|2|||REARBOX||38|F||F||PRT_EXPORT|
INS|F|R1_BOX|3|||REARBOX||38|F||F||PRT_EXPORT|
INS|F|R1_BOX|4|||REARBOX||38|F||F||PRT_EXPORT|
INS|F|R1_BOX|5|||REARBOX||38|F||F||PRT_EXPORT|
*
*   Region: R2_CAB
*
REG|FFFF|R2_CAB|0|PRT_EXPORT|
ATT|STYLE|LX
ATT|COLOR|ORANGE
ATT|A/C|YES
ATT|ENG TYPE|V6
ATT|BRAKES|POWER
INS|F|R2_CAB|1|C3294190|CT7865|CAB||35|F||F||PRT_EXPORT|
ATT|STYLE|LX
ATT|COLOR|WHITE
ATT|A/C|YES
ATT|ENG TYPE|V6
ATT|BRAKES|MANUAL
INS|F|R2_CAB|2|C4322290|CT8534|CAB||35|F||F||PRT_EXPORT|
ATT|STYLE|SX
ATT|COLOR|RED
```

```
ATT|A/C|NO
ATT|ENG TYPE|V8
ATT|BRAKES|POWER
INS|F|R2_CAB|3|C8629390|CT32119|CAB||35|F||F||PRT_EXPORT|
ATT|STYLE|LX
ATT|COLOR|BLUE
ATT|A/C|YES
ATT|ENG TYPE|V6
ATT|BRAKES|MANUAL
INS|F|R2_CAB|4|C7738990|CT7535|CAB||35|F||F||PRT_EXPORT|
ATT|STYLE|SX
ATT|COLOR|BLACK
ATT|A/C|NO
ATT|ENG TYPE|V8
ATT|BRAKES|MANUAL
INS|F|R2_CAB|5|C4296590|CT7008|CAB||35|F||F||PRT_EXPORT|
ATT|STYLE|SX
ATT|COLOR|GREEN
ATT|A/C|YES
ATT|ENG TYPE|V8
ATT|BRAKES|POWER
INS|F|R2_CAB|6|C2986590|CT0943|CAB||35|F||F||PRT_EXPORT|
ATT|STYLE|LX
ATT|COLOR|DK BLUE
ATT|A/C|YES
ATT|ENG TYPE|V6
ATT|BRAKES|MANUAL
INS|F|R2_CAB|7|C6398290|CT5476|CAB||35|F||F||PRT_EXPORT|
ATT|STYLE|SX
ATT|COLOR|YELLOW
ATT|A/C|NO
ATT|ENG TYPE|V8
ATT|BRAKES|MANUAL
INS|F|R2_CAB|8|C6884990|CT2312|CAB||35|F||F||PRT_EXPORT|
ATT|STYLE|LX
ATT|COLOR|RED/WHITE
ATT|A/C|YES
ATT|ENG TYPE|V6
ATT|BRAKES|POWER
INS|F|R2_CAB|9|C5185590|CT5643|CAB||35|F||F||PRT_EXPORT|
ATT|STYLE|LX
ATT|COLOR|GREY
ATT|A/C|YES
ATT|ENG TYPE|V6
ATT|BRAKES|POWER
```

```
INS|F|R2_CAB|10|C3299890|CT7885|CAB||35|F||F||PRT_EXPORT|
*
*   Region: R3_CAR
*
REG|FFFF|R3_CAR|0|PRT_EXPORT|
ATT|STYLE|SX
ATT|COLOR|RED
ATT|A/C|NO
ATT|ENG TYPE|V8
ATT|BRAKES|POWER
INS|F|R3_CAR|1|S2173490|ST2993|CAR||32|F||F||PRT_EXPORT|
ATT|STYLE|LX
ATT|COLOR|BLUE
ATT|A/C|YES
ATT|ENG TYPE|V8
ATT|BRAKES|MANUAL
INS|F|R3_CAR|2|S6427390|ST6991|CAR||32|F||F||PRT_EXPORT|
ATT|STYLE|SX
ATT|COLOR|GREEN
ATT|A/C|YES
ATT|ENG TYPE|V8
ATT|BRAKES|POWER
INS|F|R3_CAR|3|S4883690|ST4999|CAR||32|F||F||PRT_EXPORT|
ATT|STYLE|SX
ATT|COLOR|YELLOW
ATT|A/C|NO
ATT|ENG TYPE|V6
ATT|BRAKES|POWER
INS|F|R3_CAR|4|S1429890|ST1996|CAR||32|F||F||PRT_EXPORT|
ATT|STYLE|GTI
ATT|COLOR|GREEN
ATT|A/C|NO
ATT|ENG TYPE|TURBO
ATT|BRAKES|SELFADJUST
INS|F|R3_CAR|5|S9812390|ST9998|CAR||32|F||F||PRT_EXPORT|
```

**Note:** **`Prt_import`** does not transfer region validation statistics data.

# Production Tracking Export Program (PRT_EXPORT)

This section describes the Production Tracking Export Program (PRT_EXPORT). The tasks performed by the Export program are described, the Export file record formats are defined for both the input and output files, and a procedure for running the program is provided.

The Production Tracking Export Program (PRT_EXPORT) lets you request information from the running PRT processes, and write the information to an output file. The output file is formatted to be compatible with the PRT Import program.

## Export Input File Format

The Export Program reads in one or more records from an input file. These records specify which groups and regions of the specified service are to be dumped (all information associated with the groups and/or regions specified, and complete information on all the items within the group or region, are written to the output file).

The first field in each record in the Export file specifies the command (described below). These commands are described by denoting the literal characters to be placed in the first field of each input file record. Subsequent fields in the record are delimited by a field delimiter character (usually |) which is defined on the very first line of the Export file.

Comands are:

| Command | Function |
|---|---|
| ALL | Dumps *all* tracking data. |
| DSP | Prints to the terminal. |
| GRP | Dumps all tracking data in a specified Group. |
| REG | Dumps all tracking data in a specified Region. |
| SVC | Dumps all tracking data for a specified service or services. |

### *ALL*

Specifies that everything is to be dumped for all services. This record should be the first (and only) record in the file. If there are any subsequent records in the file, they will be ignored, i.e. processing will terminate upon completion of the complete dump of everything. If the ALL appears after some other record type, it will be ignored.

### *DSP*

Prints out the fields following the DSP field (one field per line). For example:

**DSP|Production start for item XYZ**

### *GRP*

Specifies the group identifier of a group to be dumped. This record causes all information on all regions and items within the group to be dumped. The format for the record is:

**GRP|group_identifier**

or

**GRP|group_id_1|group_id_2...**

The limit on the number of group identifiers that can be specified within a single GRP record is limited by the IDT routines that read and parse the input records. The current limits are 100 fields per record and 1000 characters per record.

### *REG*

Specifies the regions (by region identifier) which are to be dumped. The format is:

**REG|region_identifier**

or

**REG|region_id_1|region_id_2...**

The limit on the number of region identifiers that be specified within a single REG record is limited by the IDT routines that read and parse the input records. The current limits are 100 fields per record and 1000 characters per record.

### *SVC*

Specifies the service ID of the PRT service to be dumped. This record causes all information on all groups, regions, and items within the service to be dumped. The format for the record is:

**SVC|service_identifier**

or

**SVC|svc_id_1|svc_id_2...**

The limit on the number of service identifiers that can be specified within a single SVC record is limited by the IDT routines that read and parse the input records. The current limits are 100 fields per record and 1000 characters per record.

## Export Output File Format

The format for this file is identical to the format of the Import program's Input file. See the section above for the exact format of this file.

**Note:** A region may be a member of more than one group. Thus, the dumping of two different groups could include the same region twice. However, this does not result in the region being dumped each time it is encountered (either as a member of a different group or because the region was explicitly dumped).

The *first* time the region is encountered, all information on it is dumped (the region status and information on all items within the region). Subsequent encounters with the same region result in a comment record being written into the output file specifying the region name and indicating that the region had previously been dumped and appears earlier in the output file.

The same holds true for all similar circumstances, i.e. a service is dumped and then a group or region within that service is explicitly dumped (or vice versa).

**No groups or regions will be dumped more than once.**

All subsequent appearances of a group or region will be indicated by the insertion of a comment record in the output file.

## Running the Export Program

To run the Export program, first open a command prompt window. This can be done from the program manager window or a better alternative is from your CIMPLICITY HMI project window. This is better because your path to **%BSM_ROOT%** will be set automatically for you. Change directories to where you want the output file to be placed.

Then enter the following command to run the program:

**$ PRT_EXPORT <***filename***>.PRTX <***prt_system***> <***prt_node***>**

*Where*

**<***filename***>.PRTX** is an input file that specified what is to be dumped to the export file

**<***prt_system***>** is the prefix of the PRT data server you wish to export from. Specify ALL if all data servers should be connected.

**<***prt_node***>** is the node where the PRT servers are running. Specify LOCAL if they are on the same node.

If you don't specify any of the parameters on the command line, you will be prompted.

**Example Input File**

```
|-*
*
REG|R1_BOX
REG|R2_CAB
REG|R3_CAR
```

The program will create an output file called **<***filename***>.PRTM**

*Where*

**<***filename***>** is the same as that of the input file.

A sample output file is provided in the section *Running the Import Program.*

# Production Tracking Processes

This section identifies each of the processes within Production Tracking that implement the functions of the Production Tracking Subsystem. Special notes for running these processes are included.

### Production Tracking Data Collector (PRTC.EXE)

- Process ID must terminate with _DC for interprocess communication to work properly.

### Production Tracking Data Server (PRTS.EXE)

- Process ID must terminate with _DS for interprocess communication to work properly.

### Production Tracking Configuration Validation Program (PRTV.EXE)

- The CIMPLICITY HMI project for which you are checking the configuration must be running.
- This utility may be run interactively.
- Reads PRT configuration files, outputting information on a file-by-file basis about errors and/or discrepancies in configuration.

**Note:** PRT Export does not transfer region validation statistics data.

# Production Tracking Global Section Usage

This section describes the use of global sections by the PRT processes. Global sections are created by the Data Collector process for each PRT service, and are mapped onto by the corresponding Data Server Process. The backing files are located in **%SITE_ROOT%\LOG**. The following types of sections are created:

| | |
|---|---|
| <*GROUPS*>.<*PRT Service Name*> | This file is used to maintain Group hold status information. |
| <*ITEM_XREF0*>.<*PRT Service Name*> | This file is used to maintain cross-reference information about serialized items to allow fast access to items. |
| <*REGION ID*>0.<*PRT Service Name*> | This file is used to maintain item information within a particular region (there may be many of this section type per PRT service). |

*Where*

<*PRT Service Name*> is the service name of the Data Collector process and

<*REGION ID*> is a configured region identifier for this service.

It is not necessary to manually delete any of these global sections, since data integrity will then become uncertain. The exception is if changes are made to any of the following configuration files: PRT_SERVICE, PRT_REGION, PRT_GROUP, and PRT_ITEM_TYP. Global memory corruption may occur if changes are made to this configuration data and new backing files are not created when PRT is rebooted.

At startup, Production Tracking will delete the cross-reference file, ITEM_XREF0.<PRT Service Name>, then reconstruct it based upon the contents of the region files that are available. This allows the system manager to eliminate the contents of certain regions

without affecting the other regions in the service, or performing an "export-edit-import" function. To do this, the following steps must be performed:

- Terminate both the PRT data collector and server.
- Delete the appropriate region files.
- Restart the PRT data collector. This ensures that the cross-reference file will be deleted then reconstructed.
- Restart the PRT server.

**Note:** The regions are pre-allocated to a configuration-based size. If items are to be added to a region that is full, the contents of the head location are moved to the full region's detainment region. If the full region *is* a detainment region, or there is no detainment region configured for the full region, the contents of the head location are deleted.

# PRT Customization

PRT customization includes:

- Synchronization capability.
- Alarm move exceptions.

## Synchronization Capability

The auto_move_point field in the PRT_REGION file supplies the ID of a text point to be set when an item moves into a tracking region due to a "point-based" automatic move. This point is set according to the value of the NUM_SETPT_PARAM parameter in the global parameters file. The format for the NUM_SETPT_PARAM global parameter is:

**NUM_SETPT_PARAM**|**3**|<*number*>

*Where:*

- If the parameter is not present, or **<***number***>** is set to 1, the point will be set with the Item ID.
- If **<***number***>** is set to 2, the point will be set with the Item ID and Type.
- If **<***number***>** is set to 3, the point will be set with the Item ID, Type, and Reference ID.
- If **<***number***>** is set to 4, the point will be set with the Item ID, Type, Reference ID, and Source Region.

## Alarm Move Exceptions

In order to generate the "Undefined Source Region" alarm, the global parameter, **PRT_ALARM_MOVE_EXCEPTION**, must be defined in the global parameters file as follows:

```
PRT_ALARM_MOVE_EXCEPTION|1|YES
```

**Note:** Details on how to add or change a parameter in the global parameters file are found in the "*CIMPLICITY Base System User's Manual*," (GFK-1180).

# Production Tracking Application Interface

## About the Production Tracking Application Interface

CIMPLICITY HMI software's Production Tracking API lets application programs access Production Tracking functions. These functions include: adding items to the Production Tracking system; deleting items; modifying items; moving items, and modifying Region and Group tracking information. Using the API requires you to do the following:

- Understand the subroutine interfaces and system capabilities provided by CIMPLICITY HMI software's Production Tracking System. These functions are documented below.
- Understand the configuration requirements for the Production Tracking System. The Production Tracking configuration data is described in the previous chapter.
- Code appropriate applications programs.
- Compile and link the programs (as explained in this section).

## Contents of the Application Subroutine Interface

This section describes the contents of the Production Tracking Application Program Interface. These files can be used to write applications that interface with the Production Tracking system, and can also be used to compile and link a demonstration program.

The following is a list of all files distributed with the Production Tracking Application Interface utilities.

**%BSM_ROOT%** is the directory where the CIMPLICITY HMI software is installed.

### Include Files

**%BSM_ROOT%\include\PRT_API.H**

### Library Files

**%BSM_ROOT%\EXE\PRT_API.DLL**

**%BSM_ROOT%\LIB\PRT_API.LIB**

# Include Files

The following header files contain definitions used by Production Tracking procedures; they therefore must be included in an application program that interfaces with Production Tracking.

```
#include <inc_path/cor.h>
#include <inc_path/cor_stat.h>
#include <inc_path/sc_recs.h>
#include <inc_path/netcom.h>
#include <inc_path/prt_api.h>
```

(structure definitions, etc. required by the application program.)

# Run-Time Notes

If you start making off-node PRT calls to a project on a particular Tracking server, then move the project from the Tracking server to another computer, the local project will no longer be able to find the off-node project. You must restart the local project to pick up the server project's new location.

# Application Subroutines

The following application subroutines are available for your use.

## prt_api_init - Initialize The PRT_API Interface

**Name:** **prt_api_init**

**Description:**

The **PRT_API** initialization routine is responsible for initializing the interface between an application process and the **PRT_API** utilities. **prt_api_init** must be called by an application before any other **PRT_API** functions. If an application process wishes to be an interested process to PRT it must instead call one of the PRT interested process initialization routines **prt_api_register_intproc** or **prt_api_register_dyn_intproc**. If an application process wants to access the standard **PRT_API** routines as well as sign up as an interested PRT process it must call both **PRT_API** initialization routines.

**Syntax:**

```
int prt_api_init (event_flag, rcv_req_flag, prt_system, cimp_sys,
      retstat)

COR_U4      event_flag;
COR_BOOLEAN rcv_req_flag;
Char  *prt_system;
Char  *cimp_sys;
COR_STATUS  *retstat;
```

**Data Structures:**

None

**Input Arguments:**

| | |
|---|---|
| `event_flag` | Message receipt event flag; not used if `rcv_req_flag` is FALSE |
| `rcv_req_flag` | Receive requests flag; if TRUE, this indicates the application will receive requests from PRT Data Collector processes for item tracking data and serialized item hold information. |
| `prt_system` | Name of the PRT data server to connect to. This parameter allows you to connect to different Production Tracking systems with the same API. You may also specify the prefix of all the data servers you wish to connect to. This allows the API to connect to multiple servers. |
| `cimp_sys` | Name of the project or node where the PRT data collector(s) reside. |

**Output Arguments:**

| | |
|---|---|
| `*retstat` | Pointer to COR_STATUS structure |

**Return Value:**

Either **COR_SUCCESS** or **COR_FAILURE**.

If the function returns **COR_FAILURE** additional error information can be found in the **COR_STATUS** structure.

**Error Codes:**

| | |
|---|---|
| PRTI_MF_INIT_ERR | Error initializing segment (Failure) |
| PRTI_CRESEG_ERR | Error creating segment (Failure) |
| PRTI_BAD_MSG | Error in segment (Failure) |
| PRTI_FILE_OPEN_ERR | Error opening one of the configuration files (Failure) |
| PRTI_EMPTY_FILE | Configuration file has no records (Failure) |
| PRTI_FILE_READ_ERR | Error reading configuration file (Failure) |
| PRTI_REG_DEF_MISSING | Undefined Region in file (Failure) |
| PRTI_GRP_DEF_MISSING | Undefined Group in file (Failure) |
| PRTI_REGION_NOT_IN_GROUP | Region does not belong to any group (Failure) |
| PRTI_GROUP_HAS_NO_REGIONS | Group does not 'own' any regions (Failure) |
| PRTI_CFG_NO_ROUTE | Route has no source and no destination region (Failure) |
| PRTI_ITEM_TYPE_NDEF | Item type identifier not valid (Failure) |
| PRTI_ROUTE_NDEF | Non-exist. route in PRT_INV_TYPE (Failure) |
| PRTI_INV_EXPLICIT_ROUTE | Can't deny explicit item-route (Failure) |
| PRTI_NO_TYPE_FOR_ROUTE | Denied type not on route (Failure) |
| PRTI_SVC_DEF_MISSING | Undefined Service in file (Failure) |
| PRTI_BAD_SVC_NAME | Bad service name (no underscore) (Failure) |
| PRTI_XLATE_ERR | Error in **ipc_xlate** getting physical addr (Failure) |
| PRTI_AUX_NAM_ERR | Error in **ipc_aux_nam** (Failure) |
| PRTI_ADD_PORT_ERR | Error in **ipc_add_port** (Failure) |
| PRTI_WRITE_PORT_ERR | Error in **ipc_write_port** (Warn/Fail) |

## prt_api_term - Terminate PRT_API Interface

**Name:** **prt_api_term**

**Description:**

This routine is responsible for de-allocating the communication buffers allocated by the **prt_api_init** routine. If the application registered as an interested process to a PRT process it must call the **prt_api_term_intproc** routine.

**Syntax:**

```
int prt_api_term (retstat)
COR_STATUS  *retstat;
```

**Data Structures:**

None

**Input Arguments:**

None

**Output Arguments:**

| | |
|---|---|
| *retstat | Pointer to COR_STATUS structure. |

**Return Value:**

**COR_SUCCESS**

**Error Codes:**

None

# prt_api_get_req - Get Request From External Process

**Name:** **prt_api_get_req**

**Description:**

This routine provides a mechanism for an application to receive requests from the PRT Data Collector Resident Process (PRT_DC). For example, a PRT_DC can make a request to an external process for Item Tracking Data related to a specific Serialized Item. This API allows the application process to access the request issued by the PRT process.

There are currently two requests a process may receive from PRT:

- Request for Item Tracking Data - the manifest constant for this request type is **PRT_ITEM_REQ**.
- Request for instructions on placing a HOLD status on a list of serialized Items - the manifest constant for this request type is **PRT_ITEM_HOLD_REQ**.

If a Data Collector process which the application is configured to receive requests from terminates (or some other error occurs) the application will be notified by receiving a message with a request type of **PRT_ERROR_RESP**. In this case the service id of the associated Data Collector process will be placed in the prt_svc_id field of the **PRT_REQ_INFO** structure.

When **prt_api_get_req** is called, it empties the datagram receive buffer of all messages that have arrived since the last call to **prt_api_get_req** (or since the process registered as an external process) via successive reads, until a read is posted and the event flag remains low. The messages which are read are placed into one of two lists, one containing errors messages, and the other containing valid requests. If there are any messages in the error message list, these are always returned before the valid requests.

**Syntax:**

```
int prt_api_get_req (req_type, req_info, retstat)

COR_U2            *req_type;
PRT_REQ_INFO_PTR  req_info;
COR_STATUS        *retstat;
```

**Data Structures:**

```
typedef struct
{
  char        prt_svc_id[SERVICE_ID_LEN + 1] - PRT service id
  char        group_id[PRT_REGION_ID_LEN + 1] - group id
  char        item_id[PRT_ITEM_ID_LEN + 1] - item id
  char        reference_id[PRT_ITEM_ID_LEN + 1] - reference id
  COR_BOOLEAN ext_hold_active - TRUE if HOLD active
} PRT_REQ_INFO, *PRT_REQ_INFO_PTR;
```

**Input Arguments:**

None

**Output Arguments:**

| | |
|---|---|
| `*req_type` | Request type |
| `req_info` | Information contained in request |
| `*retstat` | Pointer to COR_STATUS structure |

**Return Value:**

Either COR_SUCCESS, COR_FAILURE or COR_WARNING.

If the function returns COR_WARNING or COR_FAILURE, additional error information can be found in the COR_STATUS structure.

If there are no requests pending, the function returns COR_WARNING with the COR_STATUS err_code set to PRTI_NOTHING_PENDING.

**Error Codes:**

| | |
|---|---|
| `PRTI_NOTHING_PENDING` | No requests pending (Warning) |
| `IPC_ERR_PARTDEAD` | Partner Dead (Warning) |
| `PRTI_UNKNOWN_INIT_SRC` | MF_INIT segment received from unknown service (Warning) |
| `PRTI_INIT_MSG_RCVD` | Init segment received from Data Collector (Warning) |
| `PRTI_UNEXPECTED_SEG` | Unexpected segment type (Warning) |
| `PRTI_EF_CHECK_ERR` | Error checking event flag (Failure) |
| `PRTI_BAD_MSG_MSG` | Error in segment (Failure) |
| `PRTI_READ_PORT_ERR` | Error in **`ipc_read_port`** (Failure) |

## prt_api_load_item - Load Serialized Item Tracking Data

**Name:** `prt_api_load_item`

**Description:**

This PRT_API routine is used to load the answer for a PRT request for Serialized Item Tracking Data into a response buffer. The calling routine receives back a status indicating the success or failure of the request. This function can be called multiple times to load multiple responses into the response buffer. If the buffer becomes full the return status will be unsuccessful and additional error information will be found in the COR_STATUS structure. The PRT_API routine **`prt_api_send_resp`** must be called to send the response buffer to the appropriate PRT process.

**Syntax:**

```
int prt_api_load_item (data_valid_flag, item_id,
                       reference_id, item_type_id, item_status,
                       ext_hold_active, group_id,
                       int_hold_active, int_hold_reason,
                       num_atts_valid, item_att_list, retstat)

COR_BOOLEAN   data_valid_flag;
char          item_id[PRT_ITEM_ID_LEN + 1];
char          reference_id[PRT_ITEM_ID_LEN + 1];
char          item_type_id[PRT_ITEM_TYPE_ID_LEN + 1];
COR_U4        item_status;
COR_BOOLEAN   ext_hold_active;
char          group_id[PRT_REGION_ID_LEN + 1];
COR_BOOLEAN   int_hold_active;
char          int_hold_reason[SC_DESCRIPTION_LEN + 1];
COR_I1        num_atts_valid;
PRT_ITEM_ATT  item_att_list[PRT_NUM_ATTRIBUTES];
COR_STATUS    *retstat;
```

**Data Structures:**

```
typedef struct
{
  char att_name[PRT_ATTRIBUTE_LEN + 1] - item attribute name
  char att_value[PRT_ATTRIBUTE_LEN + 1] - item attribute value

} PRT_ITEM_ATT
```

**Input Arguments:**

| | |
|---|---|
| `data_valid_flag` | If TRUE, data requested is being returned (is valid). FALSE indicates the request could not be satisfied. (required) |
| `item_id` | Unique identifier of a Serialized Item. (optional if `reference_id` is specified) |
| `reference_id` | Secondary identifier of a Serialized Item. (optional if `item_id` is specified) |
| `item_type_id` | Item type identifier. (required) |
| `item_status` | Item status. (required) |
| `ext_hold_active` | If TRUE, Serialized item held due to external hold. (optional) |

| | |
|---|---|
| group_id | Identifier of the Tracking Group placing the external hold on the item. (required if ext_hold_active = TRUE) |
| int_hold_active | If TRUE, item held due to internal hold. (optional) |
| int_hold_reason | Comment specifying reason for internal hold. (optional) |
| num_atts_valid | Number of item attributes valid; this specifies the number of array elements in the item attribute list that contain valid data. If item_att_list is a NULL pointer, the function sends a zero to the Data Collector for this argument, regardless of what was actually passed. |
| item_att_list | Item Attributes pertain to Serialized Items. (required) |

**Output Arguments:**

| | |
|---|---|
| *retstat | Pointer to COR_STATUS structure |

**Return Value:**

Either COR_SUCCESS, COR_FAILURE, or COR_WARNING.

If the function returns COR_WARNING or COR_FAILURE, additional error information can be found in the COR_STATUS structure. If no more item information will fit in the datagram buffer, the function returns COR_WARNING with the COR_STATUS err_code set to PRTI_DG_BUF_FULL.

**Error Codes:**

| | |
|---|---|
| PRTI_API_CANT_RCV | PRT Application not initialized to receive asynch messages (Warning) |
| PRTI_NON_SER_ID | Non-serialized item has item id or reference id (Warning) |
| PRTI_SER_NO_ID | Serialized item missing both item id and reference id (Warning) |
| PRTI_ITEM_TYPE_NULL | Item type identifier NULL (Warning) |
| PRTI_ITEM_TYPE_NDEF | Item type identifier not valid (Warning) |
| PRTI_GRP_ID_NULL | Group identifier NULL (Warning) |
| PRTI_GRP_ID_NDEF | Group identifier not valid (Warning) |
| PRTI_EXT_HOLD_NO_GRP | External hold specified without Group ID (Warning) |
| PRTI_INT_HOLD_NO_REASON | Internal hold specified without reason (Warning) |
| PRTI_NUM_ATTS_INVALID | Invalid number of item attributes specified (Warning) |
| PRTI_DG_BUF_FULL | Datagram buffer full (Warning) |
| PRTI_CRESEG_ERR | Error creating segment (Failure) |

# prt_api_load_item_hold - Load Serialized Item Hold Data

**Name:** `prt_api_load_item_hold`

**Description:**

This PRT_API routine is used to load the answer for a PRT request for Serialized Item Hold Data into a response buffer. The calling routine receives back a status indicating the success or failure of the request. This function can be called multiple times to load multiple responses into the response buffer. If the buffer becomes full the return status will be unsuccessful and additional error information will be found in the COR_STATUS structure. The PRT_API routine **`prt_api_send_resp`** must be called to send the response buffer to the appropriate PRT process.

**Syntax:**

```
int prt_api_load_item_hold (data_valid_flag, item_id, reference_id,
                            ext_hold_active, group_id, retstat)

COR_BOOLEAN
char         item_id[PRT_ITEM_ID_LEN + 1];
char         reference_id[PRT_ITEM_ID_LEN + 1];
COR_BOOLEAN  ext_hold_active;
char         group_id[PRT_REGION_ID_LEN + 1];
COR_STATUS   *retstat;
```

**Data Structures:**

See Static Information Requirements

**Input Arguments:**

| | |
|---|---|
| `data_valid_flag` | If TRUE, data requested is being returned (is valid). FALSE indicates the request could not be satisfied. (required) |
| `item_id` | Unique identifier of a Serialized Item. (optional if reference id specified) |
| `reference_id` | Secondary identifier of a Serialized Item. (optional if item id specified) |
| `ext_hold_active` | If TRUE, Serialized Item held due to external hold. |
| `group_id` | Tracking Group applying the hold. (required if `ext_hold_active` = TRUE) |

**Output Arguments:**

| | |
|---|---|
| `*retstat` | Pointer to COR_STATUS structure |

**Return Value:**

Either COR_SUCCESS, COR_FAILURE, or COR_WARNING.

If the function returns COR_WARNING or COR_FAILURE, additional error information can be found in the COR_STATUS structure. If no more item information will fit in the datagram buffer, the function returns COR_WARNING with the COR_STATUS err_code set to PRTI_DG_BUF_FULL.

**Error Codes:**

| | |
|---|---|
| PRTI_DG_BUF_FULL | Datagram buffer full (Warning) |
| PRTI_API_CANT_RCV | PRT Application not initialized to receive asynch messages (Warning) |
| PRTI_NON_SER_ID | Non-serialized item has item id or reference id (Warning) |
| PRTI_SER_NO_ID | Serialized item missing both item id and reference id (Warning) |
| PRTI_GRP_ID_NULL | Group identifier NULL (Warning) |
| PRTI_GRP_ID_NDEF | Group identifier not valid (Warning) |
| PRTI_EXT_HOLD_NO_GRP | External hold specified without Group ID (Warning) |
| PRTI_CRESEG_ERR | Error creating segment (Failure) |

## prt_api_send_resp - Send Response To Production Tracking

**Name:** `prt_api_send_resp`

**Description:**

This API provides a mechanism for external processes to send asynchronous responses to PRT_API requests. This routine packages the applications response and sends it to the appropriate PRT resident process.

**Syntax:**

```
int prt_api_send_resp (prt_svc_id, retstat)

char        prt_svc_id[SERVICE_ID_LEN +1];
COR_STATUS  *retstat;
```

**Data Structures:**

None

**Input Arguments:**

| | |
|---|---|
| `prt_svc_id` | PRT service identifier |

**Output Arguments:**

| | |
|---|---|
| `*retstat` | Pointer to COR_STATUS structure |

**Return Value:**

Either COR_SUCCESS or COR_FAILURE.

If the function returns COR_FAILURE additional error information can be found in the COR_STATUS structure.

**Error Codes:**

| | |
|---|---|
| `PRTI_SVC_ID_NULL` | PRT service identifier NULL (Warning) |
| `PRTI_SVC_ID_NDEF` | PRT service identifier not valid (Warning) |
| `PRTI_WRITE_PORT_ERR` | Error in **`ipc_write_port`** (Warn/Fail) |
| `PRTI_BAD_MSG` | Error in segment (Failure) |
| `PRTI_MF_INIT_ERR` | Error initializing segment (Failure) |

## prt_api_modify_item - Modify Item Tracking Data

**Name:** **`prt_api_modify_item`**

**Description:**

This routine provides a mechanism to modify Item Tracking Data, associate an Item Type to an unknown Item, associate a Reference ID to an Item ID and associate two Items. The calling routine receives back a status indicating the success or failure of the request. This routine does not allow for the modification of Item attributes, Item attributes are modified using the **`prt_api_modify_item_att`** routine.

**Syntax:**

```
int prt_api_modify_item (timestamp, modify_bitmask, status_bitmask,
                         region_id, region_loc, item_id,
                         reference_id, item_type_id,
                         parent_item_id, item_status,
                         ext_hold_active, group_id,
                         int_hold_active, int_hold_reason,
                         comment, user_or_svc_id, retstat)

time_t       timestamp;
COR_U2       modify_bitmask;
COR_U4       status_bitmask;
char         region_id[PRT_REGION_ID_LEN + 1];
COR_I2       region_loc;
char         item_id[PRT_ITEM_ID_LEN + 1];
char         reference_id[PRT_ITEM_ID_LEN + 1];
char         item_type_id[PRT_ITEM_TYPE_ID_LEN + 1];
char         parent_item_id[PRT_ITEM_ID_LEN + 1];
COR_U4       item_status;
COR_BOOLEAN  ext_hold_active;
char         group_id[PRT_REGION_ID_LEN + 1];
COR_BOOLEAN  int_hold_active;
char         int_hold_reason[SC_DESCRIPTION_LEN + 1];
char         comment[SC_DESCRIPTION_LEN + 1];
char         user_or_svc_id[SERVICE_ID_LEN + 1];
COR_STATUS   *retstat;
```

**Data Structures:**

None

**Input Arguments:** (See SYNTAX for proper order)

Serialized Items Only

| | |
|---|---|
| `item_id` | Unique identifier of a Serialized Item (optional if reference id specified) |
| `reference_id` | Secondary identifier of a Serialized Item (optional if item id specified) |
| `ext_hold_active` | If TRUE, item held due to external hold (optional) |
| `group_id` | Group identifier providing reason for hold (required if **`ext_hold_active`** = TRUE) |

Non-Serialized Items Only

| | |
|---|---|
| `parent_item_id` | Item ID of associated serialized Item (optional) |

Both

`timestamp`

The time the item was last moved or modified. The timestamp is used to ensure that, when attempting to modify an item, the item has not changed (or moved) between the time when information was last obtained for that item and the attempt to modify it. (One of the fields in the structure which returns data on an item is the last_mod_time field, which indicates the time at which the item was last moved or modified.) If the timestamp argument provided to this function is older than the timestamp associated with the item at the time of the attempted modification, then the modification is not performed and a warning returned. If the timestamp argument passed to this function is zero, the operation is performed without any timestamp checking.

`modify_bitmask`

Indicates fields that have been modified (required). If a bit in the bitmask is set, the corresponding argument provided to this function is required. If a bit is clear, the corresponding argument is not accessed in any way.

The following fields are potentially modifiable and have a corresponding bit in the **`modify_bitmask`**.

| Field Name | Bit Name |
|---|---|
| item_status | ITEM_STATUS_BIT |
| status_bitmask | ITEM_STATUS_BIT |
| item_id | ITEM_ID_MOD_BIT |
| reference_id | REFERENCE_ID_MOD_BIT |
| item_type_id | ITEM_TYPE_ID_BIT |
| parent_item_id | PARENT_ITEM_ID_BIT |
| ext_hold_active | EXT_HOLD_BIT |
| group_id | EXT_HOLD_BIT |
| int_hold_active | INT_HOLD_BIT |
| int_hold_reason | INT_HOLD_BIT |

See the next chapter for a list of macros that can be used to set the various bits in **`modify_bitmask`** to indicate which fields are being modified.

**Note:** The **`item_id`** and **`reference_id`**, if both are specified, are used to find the item unless either the ITEM_ID_MOD_BIT or REFERENCE_ID_MOD_BIT is set (both *cannot* be set). If neither is set, the item you are looking to modify will only be found if both the **`item_id`** and the **`reference_id`** you supply match that of an item already in the production tracking system. The **`item_ids`** need not match, however, if the ITEM_ID_MOD_BIT is set, and similarly, the **`reference_ids`** need not match if the REFERENCE_ID_MOD_BIT is set.

`status_bitmask`

Indicates fields in item status that have been modified. (optional)

`region_id`

Region where item last resided (required)

| | |
|---|---|
| `region_loc` | Location in the region where the item resides (required) For non-serialized items, a specific region location must be specified. For serialized items, the constant ALL_LOCATIONS may be used. This indicates that the item may reside at any location in the region. |
| `item_type_id` | Item type identifier (required for non-serialized item, optional for serialized items) |
| `item_status` | Item status (optional) |
| `comment` | Comment to be recorded in prt history log file. (optional) |
| `user_or_svc_id` | User or service identifier (optional) |
| `int_hold_active` | If TRUE, item held due to internal hold (optional) |
| `int_hold_reason` | Comment specifying reason for internal hold. (required for activate, does not apply to deactivate) |

**Output Arguments:**

| | |
|---|---|
| `*retstat` | Pointer to COR_STATUS structure |

**Return Value:**

Either COR_SUCCESS, COR_WARNING, or COR_FAILURE.

If the function returns COR_WARNING or COR_FAILURE additional error information can be found in the COR_STATUS structure.

**Error Codes:**

All error codes returnable by **`prt_api_modify_item_all`** (except PRTI_NUM_ATTS_INVALID)

# prt_api_modify_item_att - Modify Item Tracking Data Attributes

**Name:** `prt_api_modify_item_att`

**Description:**

This routine provides a mechanism to modify Serialized Item attributes. The calling routine receives back a status indicating the success or failure of the request.

**Syntax:**

```
int prt_api_modify_item_att ( timestamp, region_id, item_id,
                              reference_id, num_atts_valid,
                              item_att_list, comment,
                              user_or_svc_id, retstat)

time_t        timestamp;
char          region_id[PRT_REGION_ID_LEN + 1];
char          item_id[PRT_ITEM_ID_LEN + 1];
char          reference_id[PRT_ITEM_ID_LEN + 1];
COR_I1        num_atts_valid;
PRT_ITEM_ATT  item_att_list[PRT_NUM_ATTRIBUTES];
char          comment[SC_DESCRIPTION_LEN + 1];
char          user_or_svc_id[SERVICE_ID_LEN + 1];
COR_STATUS    *retstat;
```

**Data Structures:**

```
typedef struct
{
  char att_name[PRT_ATTRIBUTE_LEN + 1] - item attribute name
  char att_value[PRT_ATTRIBUTE_LEN + 1] - item attribute value
} PRT_ITEM_ATT
```

**Input Arguments:**

Serialized Items Only

| | |
|---|---|
| `timestamp` | The time the item was last moved or modified. The timestamp is used to ensure that, when attempting to modify an item's attributes, the item has not changed (or moved) between the time when information was last obtained for that item and the attempt to modify it. (One of the fields in the structure which returns data on an item is the last_mod_time field, which indicates the time at which the item was last moved or modified.) If the timestamp argument provided to this function is older than the timestamp associated with the item at the time of the attempted modification, then the modification is not performed and a warning returned. If the timestamp argument passed to this function is zero, the operation is performed without any timestamp checking. |
| `region_id` | Region where item last resided (required) |
| `item_id` | Unique identifier of a Serialized Item (optional if reference id specified) |
| `reference_id` | Secondary identifier of a Serialized Item (optional if item id specified) |

| | |
|---|---|
| num_atts_valid | Number of item attributes valid; this specifies the number of array elements in the item attribute list that contain valid data. |
| item_att_list | Item Attributes pertain to Serialized Items. (required) |
| comment | Comment to be recorded in PRT history log file. (optional) |
| user_or_svc_id | User or service identifier (optional) |

**Output Arguments:**

| | |
|---|---|
| *retstat | Pointer to COR_STATUS structure |

**Return Value:**

Either COR_SUCCESS, COR_WARNING, or COR_FAILURE.

If the function returns COR_WARNING or COR_FAILURE, additional error information can be found in the COR_STATUS structure.

**Error Codes:**

| | |
|---|---|
| PRTI_NUM_ATTS_INVALID | Invalid number of item attributes specified (Warning) |
| PRTC_INVALID_ITEM_TYPE | Invalid item type specified (Warning) |
| PRTC_CANT_MATCH_BOTH | Item/XREF entry mismatch (Warning) |
| PRTC_ITEM_NOT_FOUND | Item ID not found (Warning) |
| PRTC_REF_NOT_FOUND | Reference ID not found (Warning) |
| PRTC_BAD_REG_SPEC | Bad region specified (Warning) |
| PRTC_ITEM_STAMP_TOO_NEW | An affected item has changed since the last display (Warning) |
| PRTI_ITEM_TYPE_NDEF | Item type identifier not valid (Warning) |
| PRTI_SER_NO_ID | Serialized item missing both item id and reference id (Warning) |
| PRTI_REG_ID_NULL | Region identifier NULL (Warning) |
| PRTI_REG_ID_NDEF | Region identifier not valid (Warning) |
| PRTI_ITEM_TYPE_NULL | Item type identifier NULL (Warning) |
| PRTI_ITEM_TYPE_NDEF | Item type identifier not valid (Warning) |
| PRTI_MF_INIT_ERR | Error initializing segment (Failure) |
| PRTI_CRESEG_ERR | Error creating segment (Failure) |

# prt_api_modify_item_all - Modify Item Tracking Data (All)

**Name:** **`prt_api_modify_item_all`**

**Description:**

This routine provides a mechanism to modify all information fields associated with an item, both those modifiable by the **`prt_api_modify_item`** function and (for a serialized item) the item's attributes, also. For non-serialized items, this function does nothing more than the prt_api_modify_item function. The calling routine receives back a status indicating the success or failure of the request.

**Syntax:**

```
int prt_api_modify_item_all( timestamp, modify_bitmask,
                             status_bitmask, region_id,
                             region_loc, item_id, reference_id,
                             item_type_id, parent_item_id,
                             item_status, ext_hold_active,
                             group_id, int_hold_active,
                             int_hold_reason, num_atts_valid,
                             item_att_list, comment,
                             user_or_svc_id, retstat)

time_t        timestamp;
COR_U2        modify_bitmask;
COR_U4        status_bitmask;
char          region_id[PRT_REGION_ID_LEN + 1];
COR_I2        region_loc;
char          item_id[PRT_ITEM_ID_LEN + 1];
char          reference_id[PRT_ITEM_ID_LEN + 1];
char          item_type_id[PRT_ITEM_TYPE_ID_LEN + 1];
char          parent_item_id[PRT_ITEM_ID_LEN + 1];
COR_U4        item_status;
COR_BOOLEAN   ext_hold_active;
char          group_id[PRT_REGION_ID_LEN + 1];
COR_BOOLEAN   int_hold_active;
char          int_hold_reason[SC_DESCRIPTION_LEN + 1];
COR_I1        num_atts_valid;
PRT_ITEM_ATT  *item_att_list;
char          comment[SC_DESCRIPTION_LEN + 1];
char          user_or_svc_id[SERVICE_ID_LEN + 1];
COR_STATUS    *retstat;
```

**Data Structures:**

None

**Input Arguments:** (See SYNTAX for proper order)

Serialized Items Only

| | |
|---|---|
| `item_id` | Unique identifier of a Serialized Item (optional if reference id specified) |
| `reference_id` | Secondary identifier of a Serialized Item (optional if item id specified) |
| `ext_hold_active` | If TRUE, item held due to external hold (optional) |

`group_id` — Group identifier providing reason for hold (required if **`ext_hold_active`** = TRUE)

`num_atts_valid` — Number of item attributes valid; this specifies the number of array elements in the item attribute list that contain valid data.

`item_att_list` — Item Attributes pertain to Serialized Items.

Non-Serialized Items Only

`parent_item_id` — Item ID of associated serialized Item (optional)

Both

`timestamp` — The time the item was last moved or modified. The timestamp is used to ensure that, when attempting to modify an item, the item has not changed (or moved) between the time when information was last obtained for that item and the attempt to modify it. (One of the fields in the structure which returns data on an item is the last_mod_time field, which indicates the time at which the item was last moved or modified.) If the timestamp argument provided to this function is older than the timestamp associated with the item at the time of the attempted modification, then the modification is not performed and a warning returned. If the timestamp argument passed to this function is zero, the operation is performed without any timestamp checking.

`modify_bitmask` — Indicates fields that have been modified (required). If a bit in the bitmask is set, the corresponding argument provided to this function is required. If a bit is clear, the corresponding argument is not accessed in any way. The following fields are potentially modifiable and have a corresponding bit in the **`modify_bitmask`**.

| Field Name | Bit Name |
|---|---|
| item_status | ITEM_STATUS_BIT |
| status_bitmask | ITEM_STATUS_BIT |
| item_id | ITEM_ID_MOD_BIT |
| reference_id | REFERENCE_ID_MOD_BIT |
| item_type_id | ITEM_TYPE_ID_BIT |
| parent_item_id | PARENT_ITEM_ID_BIT |
| ext_hold_active | EXT_HOLD_BIT |
| group_id | EXT_HOLD_BIT |
| int_hold_active | INT_HOLD_BIT |
| int_hold_reason | INT_HOLD_BIT |
| num_atts_valid | ATTS_VALID_BIT |
| item_att_list | ATTS_VALID_BIT |

| | |
|---|---|
| | See the next chapter for a list of macros that can be used to set the various bits in **modify_bitmask** to indicate which fields are being modified.<br>Note that the **item_id** and **reference_id**, if both are specified, are used to find the item unless either the ITEM_ID_MOD_BIT or REFERENCE_ID_MOD_BIT is set (both *cannot* be set). If neither is set, the item you are looking to modify will only be found if both the **item_id** and the **reference_id** you supply match that of an item already in the production tracking system. The **item_ids** need not match, however, if the ITEM_ID_MOD_BIT is set, and similarly, the **reference_ids** need not match if the REFERENCE_ID_MOD_BIT is set. |
| status_bitmask | Indicates fields in item status that have been modified (optional) |
| region_id | Region where item last resided (required) |
| region_loc | Location in the region where the item resides (required) For non-serialized items, a specific region location must be specified. For serialized items, the constant ALL_LOCATIONS may be used. This indicates that the item may reside at any location in the region. |
| item_type_id | Item type identifier (required for non-serialized item, optional for serialized items) |
| item_status | Item status (optional) |
| comment | Comment to be recorded in PRT history log file. (optional) |
| user_or_svc_id | User or service identifier (optional) |
| int_hold_active | If TRUE, item held due to internal hold (optional) |
| int_hold_reason | Comment specifying reason for internal hold. (required for activate, does not apply to deactivate) |

**Output Arguments:**

| | |
|---|---|
| ***retstat** | Pointer to COR_STATUS structure |

**Return Value:**

Either COR_SUCCESS, COR_WARNING, or COR_FAILURE.

If the function returns COR_WARNING or COR_FAILURE additional error information can be found in the COR_STATUS structure.

**Error Codes:**

| | |
|---|---|
| PRTC_INVALID_ITEM_TYPE | Invalid item type specified (Warning) |
| PRTC_CANT_MATCH_BOTH | Item/XREF entry mismatch (Warning) |
| PRTC_ITEM_NOT_FOUND | Item ID not found (Warning) |
| PRTC_REF_NOT_FOUND | Reference ID not found (Warning) |
| PRTC_BAD_REG_SPEC | Bad region specified (Warning) |
| PRTC_BAD_REG_LOC_SPEC | Bad region location specified (Warning) |
| PRTC_ITEM_STAMP_TOO_NEW | An affected item has changed since the last display (Warning) |

| | |
|---|---|
| PRTC_NO_ITEM_AT_LOC | Specified item not found specified location (Warning) |
| PRTI_CANT_MOD_ITEM_REF_IDS | Can't modify both item & ref. ids (Warning) |
| PRTI_CANT_MOD_ITEM_WOUT_REF | Must specify ref. id to modify item id (Warning) |
| PRTI_ITEM_TYPE_NULL | Item type identifier NULL (Warning) |
| PRTI_ITEM_TYPE_NDEF | Item type identifier not valid (Warning) |
| PRTI_NON_SER_ID | Non-serialized item has item id or reference id (Warning) |
| PRTI_SER_NO_ID | Serialized item missing both item id and reference id (Warning) |
| PRTI_GRP_ID_NULL | Group identifier NULL (Warning) |
| PRTI_GRP_ID_NDEF | Group identifier not valid (Warning) |
| PRTI_EXT_HOLD_NON_SER | External hold specified for non-serialized item (Warning) |
| PRTI_INT_HOLD_NO_REASON | Internal hold specified without reason (Warning) |
| PRTI_REG_ID_NULL | Region identifier NULL (Warning) |
| PRTI_REG_ID_NDEF | Region identifier not valid (Warning) |
| PRTI_INVALID_REG_LOC | Invalid region location (Warning) |
| PRTI_ITEM_TYPE_NULL | Item type identifier NULL (Warning) |
| PRTI_ITEM_TYPE_NDEF | Item type identifier not valid (Warning) |
| PRTI_SER_PARENT | Parent ID specified for serialized item (Warning) |
| PRTI_NUM_ATTS_INVALID | Invalid number of item attributes specified (Warning) |
| PRTI_MF_INIT_ERR | Error initializing segment (Failure) |
| PRTI_CRESEG_ERR | Error creating segment (Failure) |

Plus all error codes returnable by **prt_api_send_cmd**

# prt_api_group_hold - Activate/Deactivate Hold Specification for a Tracking Group

**Name:** `prt_api_group_hold`

**Description:**

This routine provides a mechanism to activate or deactivate a HOLD specification for a Tracking Group. The calling routine receives back a status indicating the success or failure of the request.

**Syntax:**

```
int prt_api_group_hold (group_id, ext_hold_reason,
                                act_hold_flag, comment,
      user_or_svc_id,
                             retstat)
char         group_id[PRT_REGION_ID_LEN + 1];
char         ext_hold_reason[SC_DESCRIPTION_LEN + 1];
COR_BOOLEAN  act_hold_flag;
char         comment[SC_DESCRIPTION_LEN + 1];
char         user_or_svc_id[SERVICE_ID_LEN + 1];
COR_STATUS   *retstat;
```

**Data Structures:**

None

**Input Arguments:**

| | |
|---|---|
| `group_id` | Unique identifier of tracking group (required) |
| `ext_hold_reason` | Text describing the reason for placing holds on Tracking Regions in this Tracking Group (required for activate, does not apply to deactivate) |
| `act_hold_flag` | `If set to TRUE, the Hold Specification is activated. If` set to FALSE, the Hold Specification is Deactivated. (required) |
| `comment` | Comment to be recorded in PRT history log file. (optional) |
| `user_or_svc_id` | User or service identifier (optional) |

**Output Arguments:**

| | |
|---|---|
| `*retstat` | Pointer to COR_STATUS structure |

**Return Value:**

Either COR_SUCCESS, COR_WARNING, or COR_FAILURE.

If the function returns COR_WARNING or COR_FAILURE, additional error information can be found in the COR_STATUS structure.

**Error Codes:**

| | |
|---|---|
| `PRTC_NO_SUCH_GROUP` | No such group configured (Warning) |
| `PRTI_GRP_ID_NULL` | Group identifier NULL (Warning) |
| `PRTI_GRP_ID_NDEF` | Group identifier not valid (Warning) |
| `PRTI_EXT_HOLD_NO_REASON` | Group hold specified without reason (Warning) |
| `PRTI_MF_INIT_ERR` | Error initializing segment (Failure) |
| `PRTI_CRESEG_ERR` | Error creating segment (Failure) |

Plus all error codes returnable by **`prt_api_send_cmd`**

# prt_api_delete_item - Delete/Scrap Item From Tracking Region Queue

**Name:** **`prt_api_delete_item`**

**Description:**

This function provides a mechanism for deleting or scrapping a single Item from a specific Tracking Region queue. The calling routine receives back a status indicating the success or failure of the request.

**Syntax:**

```
int prt_api_delete_item ( timestamp, prt_svc_id, group_id,
                          region_id, region_loc, item_id,
                          reference_id, item_type_id,
                          disposition, comment, user_or_svc_id,
                          retstat)

time_t      timestamp;
char        prt_svc_id[SERVICE_ID_LEN + 1];
char        group_id[PRT_REGION_ID_LEN + 1];
char        region_id[PRT_REGION_ID_LEN + 1];
COR_I2      region_loc;
char        item_id[PRT_ITEM_ID_LEN + 1];
char        reference_id[PRT_ITEM_ID_LEN + 1];
char        item_type_id[PRT_ITEM_TYPE_ID_LEN + 1];
COR_I1      disposition;
char        comment[SC_DESCRIPTION_LEN + 1];
char        user_or_svc_id[SERVICE_ID_LEN + 1];
COR_STATUS  *retstat;
```

**Data Structures:**

None

**Input Arguments:** (See SYNTAX for proper order)

Serialized Items Only

`item_id` — Unique identifier of a serialized item (optional if **`reference_id`** specified)

`reference_id` — Secondary identifier of a serialized item (optional if **`item_id`** specified)

Both

`timestamp` — The time the item was last moved or modified. The timestamp is used to ensure that, when attempting to delete an item, the item has not changed (or moved) between the time when the time when information was last obtained for that item and the attempt to delete it. (One of the fields in the structure which returns data on an item is the last_mod_time field, which indicates the time at which the item was last moved or modified.) If the timestamp argument provided to this function is older than the item's current timestamp, then the deletion is not performed and a warning returned. If the timestamp argument passed to this function is zero, the operation is performed without any timestamp checking.

| | |
|---|---|
| `prt_svc_id` | PRT service identifier (optional) |
| `group_id` | Identifier of the Tracking Group (optional) |
| `region_id` | Identifier of the Tracking Region where the Item is to be deleted from the queue (required for non-serialized items, optional for serialized items)<br>**Note:** For serialized items, at least one of the above three arguments must be specified. |
| `region_loc` | Location in the region where the item resides (required). For non-serialized items, a specific region location must be specified. For serialized items, the constant ALL_LOCATIONS may be used. This indicates that the item may reside at any location in the region. |
| `item_type_id` | Item type identifier (required for non-serialized items, optional for serialized items) |
| `disposition` | Disposition of item to be deleted. (required)<br>Valid values are: |

| | |
|---|---|
| SCRAP | Causes the item to be scrapped. |
| NO_LOCK_CHECK_SCRAP | Causes the item to be scrapped, overriding region locking status. |
| DELETE | Causes the item to be deleted. |
| NO_LOCK_CHECK_DELETE | Causes the item to be deleted, overriding region locking status. |

**Note:** It may be necessary to force the delete of the item, regardless of the region's lock status, while using the PRT API functions to delete items in a region. To do this, use the "NO_LOCK_CHECK_" versions of SCRAP and DELETE above.

| | |
|---|---|
| `comment` | Comment to be recorded in PRT history log file (optional) |
| `user_or_svc_id` | User or service identifier (optional) |

**Output Arguments:**

| | |
|---|---|
| `*retstat` | Pointer to COR_STATUS structure |

**Return Value:**

Either COR_SUCCESS, COR_WARNING, or COR_FAILURE.

If the function returns COR_WARNING or COR_FAILURE, additional error information can be found in the COR_STATUS structure.

**Error Codes:**

| | |
|---|---|
| PRTC_XREF_NOFIND | Deleted, but not found in xref table (Warning) |
| PRTC_ITEM_NOT_FOUND | Item ID not found (Failure) |
| PRTC_REF_NOT_FOUND | Reference ID not found (Failure) |
| PRTC_BAD_REG_SPEC | Bad region specified (Failure) |
| PRTC_BAD_REG_LOC_SPEC | Bad region location specified (Failure) |
| PRTC_NO_ITEM_AT_LOC | Specified item not found specified location (Failure) |
| PRTC_NO_REG_LOC_SPEC | Must specify region location to delete non-serialized item (Failure) |
| PRTC_ITEM_STAMP_TOO_NEW | An affected item has changed since the last display (Failure) |
| PRTI_ITEM_TYPE_NULL | Item type identifier NULL (Warning) |
| PRTI_ITEM_TYPE_NDEF | Item type identifier not valid (Warning) |
| PRTI_NON_SER_ID | Non-serialized item has item id or reference id (Warning) |
| PRTI_SVC_ID_NULL | PRT service identifier NULL (Warning) |
| PRTI_SVC_ID_NDEF | PRT service identifier not valid (Warning) |
| PRTI_GRP_ID_NULL | Group identifier NULL (Warning) |
| PRTI_GRP_ID_NDEF | Group identifier not valid (Warning) |
| PRTI_REG_ID_NULL | Region identifier NULL (Warning) |
| PRTI_REG_ID_NDEF | Region identifier not valid (Warning) |
| PRTI_INVALID_REG_LOC | Invalid region location (Warning) |
| PRTI_INVALID_REG_LOC_MSG | Invalid region location (Warning) |
| PRTI_REG_NOT_IN_GRP | Region not in group specified (Warning) |
| PRTI_REG_NOT_IN_SVC | Region not managed by service specified (Warning) |
| PRTI_GRP_NOT_IN_SVC | Group ID inconsistent with service specified (Warning) |
| PRTI_NO_SVC_IMPLIED | No service specified (via **svc_id/group_id/region_id**) (Warning) |
| PRTI_MF_INIT_ERR | Error initializing segment (Failure) |
| PRTI_CRESEG_ERR | Error creating segment (Failure) |

Plus all error codes returnable by **prt_api_send_cmd**.

# prt_api_delete_items_at_loc - Delete/Scrap Items From Tracking Region Queue Location

**Name:** `prt_api_delete_items_at_loc`

**Description:**

This function lets users delete or scrap all Items at a location in a specific Tracking Region queue. The calling routine receives back a status indicating the success or failure of the request.

**Syntax:**

```
int prt_api_delete_items_at_loc (timestamp, region_id,
                                 region_loc, disposition,
                                 comment, user_or_svc_id,
                                 retstat)

time_t      timestamp;
char        region_id[PRT_REGION_ID_LEN + 1];
COR_I2      region_loc;
COR_I1      disposition;
char        comment[SC_DESCRIPTION_LEN + 1];
char        user_or_svc_id[SERVICE_ID_LEN + 1];
COR_STATUS  *retstat;
```

**Data Structures:**

None

**Input Arguments:** (See SYNTAX for proper order)

`timestamp`
The time the item was last moved or modified. The timestamp is used to ensure that, when attempting to delete an item, the item has not changed (or moved) between the time when the time when information was last obtained for that item and the attempt to delete it. (One of the fields in the structure that returns data on an item is the last_mod_time field, which indicates the time at which the item was last moved or modified.) If the timestamp argument provided to this function is older than the item's current timestamp, then the deletion is not performed and a warning returned. If the timestamp argument passed to this function is zero, the operation is performed without any timestamp checking.

`region_id`
Identifier of the Tracking Region where the Items are to be deleted from the queue

**Note:** For serialized items, at least one of the above three arguments must be specified.

`region_loc`
Location in the region where the items reside (required)

| | | |
|---|---|---|
| `disposition` | Disposition of item to be deleted. Valid values are: | (required) |
| | SCRAP | Causes the item to be scrapped. |
| | NO_LOCK_CHECK_SCRAP | Causes the item to be scrapped, overriding region locking status. |
| | DELETE | Causes the item to be deleted. |
| | NO_LOCK_CHECK_DELETE | Causes the item to be deleted, overriding region locking status. |
| | **Note:** It may be necessary to force the deletion of the item, regardless of the region's lock status, while using the PRT API functions to delete items in a region. To do this, use the "NO_LOCK_CHECK_" versions of SCRAP and DELETE above. | |
| `comment` | Comment to be recorded in prt history log file (optional) | |
| `user_or_svc_id` | User or service identifier (optional) | |

**Output Arguments:**

| | |
|---|---|
| `*retstat` | Pointer to COR_STATUS structure |

**Return Value:**

Either COR_SUCCESS, COR_WARNING, or COR_FAILURE.

If the function returns COR_WARNING or COR_FAILURE, additional error information can be found in the COR_STATUS structure.

**Error Codes:**

| | |
|---|---|
| `PRTC_BAD_REG_SPEC` | Bad region specified (Failure) |
| `PRTC_BAD_REG_LOC_SPEC` | Bad region location specified (Failure) |
| `PRTC_NO_ITEM_AT_LOC` | Specified item not found specified location (Failure) |
| `PRTC_NO_REG_LOC_SPEC` | Must specify region location to delete non-serialized item (Failure) |
| `PRTC_ITEM_STAMP_TOO_NEW` | An affected item has changed since the last display (Failure) |
| `PRTI_REG_ID_NULL` | Region identifier NULL (Warning) |
| `PRTI_REG_ID_NDEF` | Region identifier not valid (Warning) |
| `PRTI_INVALID_REG_LOC` | Invalid region location (Warning) |
| `PRTI_NO_SVC_IMPLIED` | No service specified (via **`region_id`**) (Warning) |
| `PRTI_MF_INIT_ERR` | Error initializing segment (Failure) |
| `PRTI_CRESEG_ERR` | Error creating segment (Failure) |

Plus all error codes returnable by **`prt_api_send_cmd`**.

## prt_api_locate_item - Locate Serialized Item

**Name:** **prt_api_locate_item**

**Description:**

This API provides a mechanism for locating a Serialized Item within all configured PRT services, a specific PRT service, a specific Tracking Group or a specific Tracking Region. If the Item is located, the Item Tracking Data is returned to the calling routine with a return status of COR_SUCCESS. If the item was located in more than one tracking region, the first location is returned in the initial call. Successive calls to this routine return the item's additional locations. When the list of locations is complete, the routine returns COR_WARNING with the additional information in the COR_STATUS structure. If the Item was not located the calling routine receives back a status indicating the request failed.

**Syntax:**

```
int prt_api_locate_item (prt_svc_id, group_id, region_id,
                         item_id, reference_id, item_data,
                         more_data, retstat)

char                 prt_svc_id[SERVICE_ID_LEN + 1];
char                 group_id[PRT_REGION_ID_LEN + 1];
char                 region_id[PRT_REGION_ID_LEN + 1];
char                 item_id[PRT_ITEM_ID_LEN + 1];
char                 reference_id[PRT_ITEM_ID_LEN + 1];
PRT_ITEM_STRUCT_PTR  item_data;
COR_BOOLEAN          *more_data;
COR_STATUS           *retstat;
```

**Data Structures:**

```
typedef struct {
  time_t - reg_entry_time
  time_t - last_mod_time
  char region_id[PRT_REGION_ID_LEN + 1] - identifier of the region
  COR_I2 region_loc - specific location in the region
  char item_id[PRT_ITEM_ID_LEN + 1]  - unique identifier of a
                                       Serialized Item.
  char reference_id[PRT_ITEM_ID_LEN + 1]  - secondary identifier
              of a Serialized Item.
  char item_type_id[PRT_ITEM_TYPE_ID_LEN + 1] - item type
                                                identifier.
  char parent_item_id[PRT_ITEM_ID_LEN + 1] - pertains only to
                                             Non-serialized Items.
  COR_U4 item_status - item status.
  COR_BOOLEAN int_hold_active - if TRUE, item held due to
                                internal hold.
  char int_hold_reason[SC_DESCRIPTION_LEN + 1]  - Comment specifying
                                                  reason for internal
                                                  hold.
  COR_BOOLEAN ext_hold_active - if TRUE, item held due to
                                external hold.
  char group_id[PRT_REGION_ID_LEN + 1] - identifier of the group
  COR_I1 num_atts_valid - Number of attributes in the list
  PRT_ITEM_ATT item_att_list[PRT_NUM_ATTRIBUTES] - Item Attributes
                                                   pertain to
                                                   Serialized
                                                   Items.
} PRT_ITEM_STRUCT, *PRT_ITEM_STRUCT_PTR
```

**Input Arguments:**

| | |
|---|---|
| `prt_svc_id` | PRT service identifier (optional) |
| `group_id` | Unique identifier of tracking group (optional) |
| `region_id` | Unique identifier of tracking region (optional) |
| `item_id` | Unique identifier of serialized item (optional if **`reference_id`** specified) |
| `reference_id` | Secondary identifier of serialized item (optional if **`item_id`** specified) |
| `more_data` | If this flag is set to true by the caller, the function attempts to return an item found by a previous call to this function. If this flag is set to FALSE, any items found by a previous call to this function are deleted and new request(s) are made to the specified service(s). |

**Output Arguments:**

| | |
|---|---|
| `PRT_ITEM_STRUCT_PTR item_data` | Serialized Item Tracking Data |
| `more_data` | If only a single item was found, this flag is set to FALSE. If more than one item was found, this flag is set to TRUE. |
| `*retstat` | Pointer to COR_STATUS structure |

**Return Value:**

Either COR_SUCCESS, COR_WARNING, or COR_FAILURE.

If the function returns COR_WARNING or COR_FAILURE, additional error information can be found in the COR_STATUS structure.

**Error Codes:**

| | |
|---|---|
| `PRTS_NO_GLOBAL_XREF` | No XREF global section available (Warning) |
| `PRTS_REGION_NOT_DEFINED` | Region not configured (Warning) |
| `PRTS_NO_GLOBAL_REGION` | No global section available for Region (Warning) |
| `PRTI_BAD_POINTER` | Pointer to data structure is NULL (Warning) |
| `PRTI_NO_MORE_ITEMS` | No more items available (Warning) |
| `PRTI_SVC_ID_NULL` | PRT service identifier NULL (Warning) |
| `PRTI_SVC_ID_NDEF` | PRT service identifier not valid (Warning) |
| `PRTI_REG_ID_NULL` | Region identifier NULL (Warning) |
| `PRTI_REG_ID_NDEF` | Region identifier not valid (Warning) |
| `PRTI_GRP_ID_NULL` | Group identifier NULL (Warning) |
| `PRTI_GRP_ID_NDEF` | Group identifier not valid (Warning) |
| `PRTI_NON_SER_ID` | Non-serialized item has **`item_id`** or **`reference_id`** (Warning) |
| `PRTI_SER_NO_ID` | Serialized item missing both **`item_id`** and **`reference_id`** (Warning) |
| `PRTI_REG_NOT_IN_GRP` | Region not in group specified (Warning) |
| `PRTI_REG_NOT_IN_SVC` | Region not managed by service specified (Warning) |

| | |
|---|---|
| PRTI_GRP_NOT_IN_SVC | Group ID inconsistent with service specified (Warning) |
| PRTI_UNEXPECTED_SEG | Unexpected segment type (Warning) |
| PRTI_ITEM_NOT_FOUND_INC_SRCH | Item not found: some service(s) down (Warning) |
| PRTI_ITEM_NOT_FOUND | Item not found (Warning) |
| PRTI_WRITE_PORT_ERR | Error in **ipc_write_port** (Warn/Fail) |
| PRTI_MF_INIT_ERR | Error initializing segment (Failure) |
| PRTI_CRESEG_ERR | Error creating segment (Failure) |
| PRTI_BAD_MSG | Error in segment (Failure) |
| PRTI_BAD_SERVER_RESPONSE | PRT Data Server returned bad/error response (Failure) |
| PRTI_UNKNOWN_RESPONDER | Response received from unknown service (Failure) |

# prt_api_req_groupitemlist - Request List Of Serialized Item IDs in a Tracking Group

**Name:** `prt_api_req_groupitemlist`

**Description:**

This routine provides a mechanism to request all Serialized Item IDs in a Tracking Group. If the request is a success the calling routine will receive back a count of the number of items in the list of Serialized Items in the Group. If the request fails, the calling routine will receive back status information on the reason for the failure.

**Syntax:**

```
prt_api_req_groupitemlist (group_id, item_count, retstat)

char        group_id[PRT_REGION_ID_LEN + 1];
COR_I4      *item_count;
COR_STATUS  *retstat;
```

**Data Structures:**

None

**Input Arguments:**

| | |
|---|---|
| `group_id` | Unique identifier of tracking group (required) |

**Output Arguments:**

| | |
|---|---|
| `*item_count` | `Count of Items returned to schema` |
| `*retstat` | Pointer to COR_STATUS structure |

**Return Value:**

Either COR_SUCCESS, COR_WARNING or COR_FAILURE.

If the function returns COR_WARNING or COR_FAILURE, additional error information can be found in the COR_STATUS structure.

**Error Codes:**

| | |
|---|---|
| `PRTS_NO_GLOBAL_XREF` | No XREF global section available (Warning) |
| `PRTS_REGION_NOT_DEFINED` | Region not configured (Warning) |
| `PRTS_NO_GLOBAL_REGION` | No global section available for Region (Warning) |
| `PRTS_NO_ITEMS_AT_LOCATION` | There are no items at location specified (Warning) |
| `PRTI_GRP_ID_NULL` | Group identifier NULL (Warning) |
| `PRTI_GRP_ID_NDEF` | Group identifier not valid (Warning) |
| `PRTI_UNEXPECTED_SEG_TYPE` | Unexpected segment type from PRT_DS (Warning) |
| `PRTI_PARTNER_DEAD` | Partner dead (Warning) |
| `PRTI_EXTRA_SEGS` | Ignored extra segments after STATUS_SEG or END_SEG (Warning) |
| `PRTI_WRITE_PORT_ERR` | Error in **`ipc_write_port`** (Warn/Fail) |
| `PRTI_MF_INIT_ERR` | Error initializing segment (Failure) |
| `PRTI_CRESEG_ERR` | Error creating segment (Failure) |
| `PRTI_BAD_MSG` | Error in segment (Failure) |
| `PRTI_READ_PORT_ERR` | Error in **`ipc_read_port`** (Failure) |

# prt_api_unload_groupitemlist - Unload List Of Serialized Item IDs in a Tracking Group

**Name:** **prt_api_unload_groupitemlist**

**Description::**

This routine provides a mechanism to unload the list of Serialized Items obtained through the **prt_api_req_groupitemlist** routine. If the request is a success the calling routine will receive back an array populated with the Serialized Item information. If the request fails, the calling routine will receive back status information on the reason for the failure.

**Syntax:**

```
int prt_api_unload_groupitemlist (list_size, item_hold_list,
                                  retstat)

COR_I4                list_size;
PRT_ITEM_HOLD_STRUCT  item_hold_list[list_size];
COR_STATUS            *retstat;
```

**Data Structures:**

```
typedef struct {
  char item_id[PRT_ITEM_ID_LEN + 1] - unique identifier of
                                     the Item.
  char group_id[PRT_REGION_ID_LEN + 1] - identifier of group
                                         responsible for
                                         placing a hold on this item
  COR_BOOLEAN ext_hold_active - if TRUE, HOLD is activated
                                         for Item.
} PRT_ITEM_HOLD_STRUCT, *PRT_ITEM_HOLD_STRUCT_PTR;
```

**Input Arguments:**

| | |
|---|---|
| list_size | Size allocated for **item_hold_list** array (required) |
| item_hold_list | Array of hold items (required) |

**Output Arguments:**

| | |
|---|---|
| item_hold_list | Populated array of hold items |
| *retstat | Pointer to COR_STATUS structure |

**Return Value:**

Either COR_SUCCESS, COR_FAILURE or COR_WARNING.

If the function returns COR_FAILURE additional error information can be found in the COR_STATUS structure.

**Error Codes:**

| | |
|---|---|
| PRTI_TOO_MANY_REQ | Total number in list less than number requested (Warning) |

# prt_api_req_class_itemlist - Request List Of Item Tracking Data

**Name:** `prt_api_req_class_itemlist`

**Description::**

This routine provides a mechanism to request the Item Tracking Data for one or more Items of a particular Item Class from the PRT Data Server (PRT_DS) residing within all configured PRT services, a specific PRT service, a specific Tracking Group, a specific Tracking Region, or at a specific location within a specific Tracking Region. If the request is a success the calling routine will receive back a count of the number of items returned to the item list. If the request fails, the calling routine will receive back status information on the reason for the failure.

**Syntax:**

```
int prt_api_req_class_itemlist (prt_svc_id, group_id,
                                region_id, region_loc,
                                item_class_id, item_count,
                                max_location, retstat)

char        prt_svc_id[SERVICE_ID_LEN + 1];
char        group_id[PRT_REGION_ID_LEN + 1];
char        region_id[PRT_REGION_ID_LEN + 1];
COR_I2      region_loc;
char        item_class_id[PRT_ITEM_CLASS_ID_LEN+1];
COR_I4      *item_count;
COR_I4      *max_location;
COR_STATUS  *retstat;
```

**Data Structures:**

None

**Input Arguments:**

| | |
|---|---|
| `prt_svc_id` | PRT service identifier (optional, but required if no other arguments specified) |
| `group_id` | Identifier of the group. (optional, but required if no other arguments specified) |
| `region_id` | Identifier of the region. (optional, but required if no other arguments specified, required if region location specified) |

**Note:** At least one of the above three arguments must be specified. If not, all services will be queried. It is possible that a list of 'items' will be collected, but not all services will have responded (because one or more of them were down). Thus the user has a list of valid responses, but may *not* have ALL the 'items' that should have been returned. In this case, the function will return COR_WARNING with an `err_code` of PRTI_NOT_ALL_SVCS_RESPONDED. The function returns, through an argument passed, the number of 'items' actually received. This argument will show how many valid 'items' were received.

| | |
|---|---|
| region_loc | Specific location in the region (required). If all items within a region are being requested, the **region_loc** must be specified as ALL_LOCATIONS. Otherwise a specific location requests items at only that location. |
| Item_class_id | Filter to use to select a particular class. Enter a blank to select all items (that is for no filtering). |

**Output Arguments:**

| | |
|---|---|
| *item_count | Count of Items returned to schema |
| *max_location | Location number of the highest populated location in the region, regardless of the Item Class in the region. |
| *retstat | Pointer to COR_STATUS structure |

**Return Value:**

Either COR_SUCCESS, COR_WARNING or COR_FAILURE.

If the function returns COR_WARNING or COR_FAILURE, additional error information can be found in the COR_STATUS structure.

**Error Codes:**

| | |
|---|---|
| PRTS_NO_GLOBAL_XREF | No XREF global section available (Warning) |
| PRTS_REGION_NOT_DEFINED | Region not configured (Warning) |
| PRTS_NO_GLOBAL_REGION | No global section available for Region (Warning) |
| PRTS_NO_ITEMS_AT_LOCATION | There are no items at location specified (Warning) |
| PRTI_SVC_ID_NULL | PRT service identifier NULL (Warning) |
| PRTI_SVC_ID_NDEF | PRT service identifier not valid (Warning) |
| PRTI_REG_ID_NULL | Region identifier NULL (Warning) |
| PRTI_REG_ID_NDEF | Region identifier not valid (Warning) |
| PRTI_INVALID_REG_LOC | Invalid region location (Warning) |
| PRTI_GRP_ID_NULL | Group identifier NULL (Warning) |
| PRTI_GRP_ID_NDEF | Group identifier not valid (Warning) |
| PRTI_REG_NOT_IN_GRP | Region not in group specified (Warning) |
| PRTI_REG_NOT_IN_SVC | Region not managed by service specified (Warning) |
| PRTI_GRP_NOT_IN_SVC | Group ID inconsistent with service specified (Warning) |
| PRTI_UNEXPECTED_SEG_TYPE | Unexpected segment type from PRT_DS (Warning) |
| PRTI_PARTNER_DEAD | Partner dead (Warning) |
| PRTI_EXTRA_SEGS | Ignored extra segments after STATUS_SEG or END_SEG (Warning) |
| PRTI_NOT_ALL_SVCS_RESPONDED | Not all services responded (Warning) |
| PRTI_WRITE_PORT_ERR | Error in **ipc_write_port** (Warn/Fail) |
| PRTI_MF_INIT_ERR | Error initializing segment (Failure) |
| PRTI_CRESEG_ERR | Error creating segment (Failure) |
| PRTI_BAD_MSG | Error in segment (Failure) |
| PRTI_READ_PORT_ERR | Error in **ipc_read_port** (Failure) |

# prt_api_req_itemlist - Request List Of Item Tracking Data

**Name:** `prt_api_req_itemlist`

**Description::**

This routine provides a mechanism to request the Item Tracking Data for one or more Items from the PRT Data Server (PRT_DS) residing within all configured PRT services, a specific PRT service, a specific Tracking Group, a specific Tracking Region, or at a specific location within a specific Tracking Region. If the request is a success the calling routine will receive back a count of the number of items returned to the item list. If the request fails, the calling routine will receive back status information on the reason for the failure.

**Syntax:**

```
int prt_api_req_itemlist (prt_svc_id, group_id, region_id,
                          region_loc, item_count, retstat)

char        prt_svc_id[SERVICE_ID_LEN + 1];
char        group_id[PRT_REGION_ID_LEN + 1];
char        region_id[PRT_REGION_ID_LEN + 1];
COR_I2      region_loc;
COR_I4      *item_count;
COR_STATUS  *retstat;
```

**Data Structures:**

None

**Input Arguments:**

| | |
|---|---|
| `prt_svc_id` | PRT service identifier (optional, but required if no other arguments specified) |
| `group_id` | Identifier of the group. (optional, but required if no other arguments specified) |
| `region_id` | Identifier of the region. (optional, but required if no other arguments specified, required if region location specified)<br><br>**Note:** At least one of the above three arguments must be specified. If not, all services will be queried. It is possible that a list of 'items' will be collected, but not all services will have responded (because one or more of them were down). Thus the user has a list of valid responses, but may *not* have *all* the 'items' that should have been returned. In this case, the function will return COR_WARNING with an `err_code` of PRTI_NOT_ALL_SVCS_RESPONDED. The function returns, through an argument passed, the number of 'items' actually received. This argument will show how many valid 'items' were received. |
| `region_loc` | Specific location in the region (required). If all items within a region are being requested, the **`region_loc`** must be specified as ALL_LOCATIONS. Otherwise a specific location requests items at only that location. |

**Output Arguments:**

| | |
|---|---|
| `*item_count` | Count of Items returned to schema |
| `*retstat` | Pointer to COR_STATUS structure |

**Return Value:**

Either COR_SUCCESS, COR_WARNING or COR_FAILURE.

If the function returns COR_WARNING or COR_FAILURE, additional error information can be found in the COR_STATUS structure.

**Error Codes:**

| | |
|---|---|
| `PRTS_NO_GLOBAL_XREF` | No XREF global section available (Warning) |
| `PRTS_REGION_NOT_DEFINED` | Region not configured (Warning) |
| `PRTS_NO_GLOBAL_REGION` | No global section available for Region (Warning) |
| `PRTS_NO_ITEMS_AT_LOCATION` | There are no items at location specified (Warning) |
| `PRTI_SVC_ID_NULL` | PRT service identifier NULL (Warning) |
| `PRTI_SVC_ID_NDEF` | PRT service identifier not valid (Warning) |
| `PRTI_REG_ID_NULL` | Region identifier NULL (Warning) |
| `PRTI_REG_ID_NDEF` | Region identifier not valid (Warning) |
| `PRTI_INVALID_REG_LOC` | Invalid region location (Warning) |
| `PRTI_GRP_ID_NULL` | Group identifier NULL (Warning) |
| `PRTI_GRP_ID_NDEF` | Group identifier not valid (Warning) |
| `PRTI_REG_NOT_IN_GRP` | Region not in group specified (Warning) |
| `PRTI_REG_NOT_IN_SVC` | Region not managed by service specified (Warning) |
| `PRTI_GRP_NOT_IN_SVC` | Group ID inconsistent with service specified (Warning) |
| `PRTI_UNEXPECTED_SEG_TYPE` | Unexpected segment type from PRT_DS (Warning) |
| `PRTI_PARTNER_DEAD` | Partner dead (Warning) |
| `PRTI_EXTRA_SEGS` | Ignored extra segments after STATUS_SEG or END_SEG (Warning) |
| `PRTI_NOT_ALL_SVCS_RESPONDED` | Not all services responded (Warning) |
| `PRTI_WRITE_PORT_ERR` | Error in **`ipc_write_port`** (Warn/Fail) |
| `PRTI_MF_INIT_ERR` | Error initializing segment (Failure) |
| `PRTI_CRESEG_ERR` | Error creating segment (Failure) |
| `PRTI_BAD_MSG` | Error in segment (Failure) |
| `PRTI_READ_PORT_ERR` | Error in **`ipc_read_port`** (Failure) |

## prt_api_unload_itemlist - Unload List Of Items

**Name:** **`prt_api_unload_itemlist`**

**Description::**

This routine provides a mechanism to unload the list of Items obtained through the **`prt_api_req_itemlist`** routine. If the request is a success the calling routine will receive back an array populated with the Item Tracking Data information. If the request fails, the calling routine will receive back status information on the reason for the failure.

**Syntax:**

```
int prt_api_unload_itemlist (list_size, item_list, retstat)

COR_I4           list_size;
PRT_ITEM_STRUCT  item_list[list_size];
COR_STATUS       *retstat;
```

**Data Structures:**

```
typedef struct {
  time_t - reg_entry_time
  time_t - last_mod_time
  char region_id[PRT_REGION_ID_LEN + 1] - identifier of the region
  COR_I2 region_loc - specific location in the region
  char item_id[PRT_ITEM_ID_LEN + 1] - unique identifier of a
                                      Serialized Item.
  char reference_id[PRT_ITEM_ID_LEN + 1]  - secondary identifier
                                            of a Serialized Item.
  char item_type_id[PRT_ITEM_TYPE_ID_LEN + 1] - item type identifier.
  char parent_item_id[PRT_ITEM_ID_LEN + 1] - pertains only to
                                             Non-serialized Items.
  COR_U4 item_status - item status.
  COR_BOOLEAN int_hold_active - if TRUE, item held due to
                                internal hold.
  char int_hold_reason[SC_DESCRIPTION_LEN + 1]  - Comment
                       specifying reason for internal hold.
  COR_BOOLEAN ext_hold_active - if TRUE, item held due to
                                external hold.
  char group_id[PRT_REGION_ID_LEN + 1] - identifier of the group
  COR_I1 num_atts_valid - Number of valid attributes in the list
  PRT_ITEM_ATT item_att_list[PRT_NUM_ATTRIBUTES] - Item Attributes
                             pertain to Serialized Items.
} PRT_ITEM_STRUCT, *PRT_ITEM_STRUCT_PTR
```

**Input Arguments:**

| | |
|---|---|
| `list_size` | Number of array elements allocated for **`item_list`** array (required) |
| `item_list` | Pointer to a buffer to receive returned items (required) |

**Output Arguments:**

| | |
|---|---|
| `item_list` | Populated list of Items |
| `*retstat` | Pointer to COR_STATUS structure |

**Return Value:**

Either COR_SUCCESS, COR_FAILURE or COR_WARNING.

If the function returns COR_FAILURE additional error information can be found in the COR_STATUS structure.

**Error Codes:**

| | |
|---|---|
| PRTI_TOO_MANY_REQ | Total number in list less than number requested (Warning) |

# prt_api_req_regionlist - Request List Of Tracking Region Data

**Name:** `prt_api_req_regionlist`

**Description:**

This routine provides a mechanism to request Tracking Region Data for one or more Tracking Regions in a Tracking Region Group residing within all configured PRT services, a specific PRT service, a specific Tracking Group or a specific Tracking Region. If the request is a success the calling routine will receive back a count of the number of Tracking Regions returned to the region list. If the request fails, the calling routine will receive back status information on the reason for the failure.

**Syntax:**

```
prt_api_req_regionlist (prt_svc_id, group_id, region_id,
                        item_count, retstat)

char        prt_svc_id[SERVICE_ID_LEN + 1];
char        group_id[PRT_REGION_ID_LEN + 1];
char        region_id[PRT_REGION_ID_LEN + 1];
COR_I4      *item_count;
COR_STATUS  *retstat;
```

**Data Structures:**

None

**Input Arguments:**

| | |
|---|---|
| `prt_svc_id` | PRT service identifier (optional, but required if no other arguments specified) |
| `group_id` | Tracking group identifier (optional, but required if no other arguments specified) |
| `region_id` | Tracking region identifier (optional, but required if no other arguments specified) |

**Note:** At least one of the above three arguments must be specified. If not, all services will be queried. It is possible that a list of 'items' will be collected, but not all services will have responded (because one or more of them were down). Thus the user has a list of valid responses, but may *not* have *all* the 'items' that should have been returned. In this case, the function will return COR_WARNING with an **`err_code`** of PRTI_NOT_ALL_SVCS_RESPONDED. The function returns, through an argument passed, the number of 'items' actually received. This argument will show how many valid 'items' were received.

**Output Arguments:**

| | |
|---|---|
| `*item_count` | Count of regions returned to schema |
| `*retstat` | Pointer to COR_STATUS structure |

**Return Value:**

Either COR_SUCCESS, COR_WARNING or COR_FAILURE.

If the function returns COR_WARNING or COR_FAILURE, additional error information can be found in the COR_STATUS structure.

**Error Codes:**

| | |
|---|---|
| PRTS_REGION_NOT_DEFINED | Region not configured (Warning) |
| PRTS_GROUP_NOT_DEFINED | Group not configured (Warning) |
| PRTS_NO_REGIONS_MAPPED | No regions attempted were mapped (Warning) |
| PRTS_NO_INFO | No information available due to unknown reason (Warning) |
| PRTS_NO_GLOBAL_REGION | No global section available for Region (Warning) |
| PRTI_SVC_ID_NULL | PRT service identifier NULL (Warning) |
| PRTI_SVC_ID_NDEF | PRT service identifier not valid (Warning) |
| PRTI_REG_ID_NULL | Region identifier NULL (Warning) |
| PRTI_REG_ID_NDEF | Region identifier not valid (Warning) |
| PRTI_GRP_ID_NULL | Group identifier NULL (Warning) |
| PRTI_GRP_ID_NDEF | Group identifier not valid (Warning) |
| PRTI_REG_NOT_IN_GRP | Region not in group specified (Warning) |
| PRTI_REG_NOT_IN_SVC | Region not managed by service specified (Warning) |
| PRTI_GRP_NOT_IN_SVC | Group ID inconsistent with service specified (Warning) |
| PRTI_UNEXPECTED_SEG_TYPE | Unexpected segment type from PRT_DS (Warning) |
| PRTI_PARTNER_DEAD | Partner dead (Warning) |
| PRTI_EXTRA_SEGS | Ignored extra segments after STATUS_SEG or END_SEG (Warning) |
| PRTI_NOT_ALL_SVCS_RESPONDED | Not all services responded (Warning) |
| PRTI_WRITE_PORT_ERR | Error in **ipc_write_port** (Warn/Fail) |
| PRTI_MF_INIT_ERR | Error initializing segment (Failure) |
| PRTI_CRESEG_ERR | Error creating segment (Failure) |
| PRTI_BAD_MSG | Error in segment (Failure) |
| PRTI_READ_PORT_ERR | Error in **ipc_read_port** (Failure) |

# prt_api_unload_regionlist - Unload List Of Tracking Region Data

**Name:** **prt_api_unload_regionlist**

**Description:**

This routine provides a mechanism to unload the list of Tracking Regions obtained through the **prt_api_req_regionlist** routine. If the request is a success the calling routine will receive back an array populated with the Tracking Region Data. This array contains a pointer to another array (allocated by the application interface routine) which contains an element for each type of item in the region and the total number of items of that type in the region. The caller is responsible for de-allocating the PRT_ITEM_COUNTS array when it is no longer needed. If the request fails, the calling routine will receive back status information on the reason for the failure.

**Syntax:**

```
int prt_api_unload_regionlist (list_size, region_list, retstat)

COR_I4             list_size;
PRT_REGION_STRUCT  region_list[list_size];
COR_STATUS         *retstat;
```

**Data Structures:**

```
typedef struct {
  COR_I4 item_type_count - number of items of that type in region
  char   item_type_id[PRT_ITEM_TYPE_ID_LEN] - item type identifier
} PRT_ITEM_COUNTS;

typedef struct {
  char   region_id[PRT_REGION_ID_LEN + 1] - unique identifier
                                           of tracking region.
  COR_U4 region_status - region status.
  COR_I4 total_items - total number of items in region
  COR_I2 region_cap - number of item carriers in region
  char   stat_ptid[POINT_ID_LEN + 1] - status point identifier
  char   item_qty_ptid[POINT_ID_LEN + 1] - item quantity point id
  COR_I2 total_item_counts - tells how many PRT_ITEM_COUNTS
                             elements are in the array pointed
                             to by type_totals.
  PRT_ITEM_COUNTS *type_totals - pointer to array of PRT_ITEM_COUNTS
                                 structures (array must be
                                 deallocated by the user)
} PRT_REGION_STRUCT, *PRT_REGION_STRUCT_PTR;
```

**Input Arguments:**

| | |
|---|---|
| `list_size` | number of array elements allocated for **region_list** array (required) |
| `region_list` | Pointer to a buffer to receive tracking region data (required) |

**Output Arguments:**

| | |
|---|---|
| `region_list` | populated array of Tracking Region Data |
| `*retstat` | pointer to COR_STATUS structure |

**Return Value:**

Either COR_SUCCESS, COR_WARNING or COR_FAILURE.

If the function returns COR_WARNING or COR_FAILURE, additional error information can be found in the COR_STATUS structure.

**Error Codes:**

| | |
|---|---|
| PRTI_TOO_MANY_REQ | Total number in list less than number requested (Warning) |

The following code example shows how to deallocate the user memory:

```
PRT_REGION_STRUCT region_struct, *region_struct_ptr;
COR_STATUS retstat;

if (prt_api_unload_regionlist (1, &region_struct, &retstat) ==
      COR_SUCCESS)
{
  region_struct_ptr = &region_struct;

  /* Deallocate PRT_ITEM_COUNTS structure */
  if ( region_struct_ptr->total_item_counts > 0 )
  {
    cor_mfree ( region_struct_ptr->type_totals, 0 );
  }
}
```

# prt_api_req_grouplist - Request List Of Tracking Group Data

**Name:** `prt_api_req_grouplist`

**Description:**

This routine provides a mechanism to request Tracking Group Data for one or more Tracking Groups residing within all configured PRT services, a specific PRT service or a specific Tracking Group. If the request is a success the calling routine will receive back a count of the number of Tracking Groups returned to the group list. If the request fails, the calling routine will receive back status information on the reason for the failure.

**Syntax:**

```
int prt_api_req_grouplist (prt_svc_id, group_id, group_count, retstat)

char        prt_svc_id[SERVICE_ID_LEN + 1];
char        group_id[PRT_REGION_ID_LEN + 1];
COR_I4      *group_count;
COR_STATUS  *retstat;
```

**Data Structures:**

None

**Input Arguments:**

| | |
|---|---|
| `prt_svc_id` | Service identifier (optional, if **`group_id`** is specified) |
| `group_id` | Tracking group identifier (optional, if **`prt_svc_id`** is specified) |

**Note:** At least one of the above arguments must be specified. If not, all services will be queried. It is possible that a list of 'items' will be collected, but not all services will have responded (because one or more of them were down). Thus the user has a list of valid responses, but may *not* have *all* the 'items' that should have been returned. In this case, the function will return COR_WARNING with an **`err_code`** of PRTI_NOT_ALL_SVCS_RESPONDED. The function returns, through an argument passed, the number of 'items' actually received. This argument will show how many valid 'items' were received.

**Output Arguments:**

| | |
|---|---|
| `*group_count` | Count of Groups returned to schema |
| `*retstat` | Pointer to COR_STATUS structure |

**Return Value:**

Either COR_SUCCESS, COR_WARNING or COR_FAILURE.

If the function returns COR_WARNING or COR_FAILURE, additional error information can be found in the COR_STATUS structure.

**Error Codes:**

| | |
|---|---|
| PRTS_GROUP_NOT_DEFINED | Group not configured (Warning) |
| PRTS_NO_INFO | No information available due to unknown reason (Warning) |
| PRTS_NO_GLOBAL_GROUP | No GROUPS global section available for (Warning) |
| PRTI_SVC_ID_NULL | PRT service identifier NULL (Warning) |
| PRTI_SVC_ID_NDEF | PRT service identifier not valid (Warning) |
| PRTI_GRP_ID_NULL | Group identifier NULL (Warning) |
| PRTI_GRP_ID_NDEF | Group identifier not valid (Warning) |
| PRTI_GRP_NOT_IN_SVC | Group ID inconsistent with service specified (Warning) |
| PRTI_UNEXPECTED_SEG_TYPE | Unexpected segment type from PRT_DS (Warning) |
| PRTI_PARTNER_DEAD | Partner dead (Warning) |
| PRTI_EXTRA_SEGS | Ignored extra segments after STATUS_SEG or END_SEG (Warning) |
| PRTI_NOT_ALL_SVCS_RESPONDED | Not all services responded (Warning) |
| PRTI_WRITE_PORT_ERR | Error in **ipc_write_port** (Warn/Fail) |
| PRTI_MF_INIT_ERR | Error initializing segment (Failure) |
| PRTI_CRESEG_ERR | Error creating segment (Failure) |
| PRTI_BAD_MSG | Error in segment (Failure) |
| PRTI_READ_PORT_ERR | Error in **ipc_read_port** (Failure) |

# prt_api_unload_grouplist - Unload List Of Tracking Groups

**Name:** **prt_api_unload_grouplist**

**Description:**

This routine provides a mechanism to unload the list of Tracking Groups obtained through the **prt_api_req_grouplist** routine. If the request is a success the calling routine will receive back an array populated with the Tracking Group Data. If the request fails, the calling routine will receive back status information on the reason for the failure.

**Syntax:**

```
int prt_api_unload_grouplist (list_size, group_list, retstat)

COR_I4            list_size;
PRT_GROUP_STRUCT  group_list[list_size];
COR_STATUS        *retstat;
```

**Data Structures:**

```
typedef struct {
  char group_id[PRT_REGION_ID_LEN + 1] - unique identifier of
                                         tracking group.
  COR_BOOLEAN ext_hold_active - TRUE indicates HOLD is active
                                for group.
  char ext_hold_reason[SC_DESCRIPTION_LEN + 1] - reason for
                                                 Item HOLDS.
} PRT_GROUP_STRUCT, * PRT_GROUP_STRUCT_PTR;
```

**Input Arguments:**

| | |
|---|---|
| list_size | Number of array elements allocated for **region_list** array (required) |
| region_list | Pointer to a buffer to receive tracking region data (required) |

**Output Arguments:**

| | |
|---|---|
| group_list[list_size] | Populated list of groups |
| *retstat | Pointer to COR_STATUS structure |

**Return Value:**

Either COR_SUCCESS, COR_WARNING or COR_FAILURE.

If the function returns COR_WARNING or COR_FAILURE, additional error information can be found in the COR_STATUS structure.

**Error Codes:**

| | |
|---|---|
| PRTI_TOO_MANY_REQ | Total number in list less than number requested (Warning) |

# prt_api_register_intproc - Register As An Interested Process

**Name:** **prt_api_register_intproc**

**Description:**

This function provides a mechanism for a process to request to register as an interested process to a PRT process. Either this initialization routine must be called by all PRT interested processes prior to calling any other PRT_API interested process utilities or **prt_api_dyn_register_intproc**. The calling routine receives back a status indicating the success or failure of the request. This routine is used when you have configured your process as a service and is included in the **prt_service.dat** file. If this is not the case then you must use prt_api_dyn_register_intproc.

**Syntax:**

```
int prt_api_register_intproc (intproc_event_flag, retstat)

COR_I4      intproc_event_flag;
COR_STATUS  *retstat;
```

**Data Structures:**

None

**Input Arguments:**

| | |
|---|---|
| intproc_event_flag | Interested process event flag (required) |

**Output Arguments:**

| | |
|---|---|
| ***retstat** | Pointer to COR_STATUS structure |

**Return Value:**

Either COR_SUCCESS or COR_FAILURE.

If the function returns COR_FAILURE additional error information can be found in the COR_STATUS structure.

**Error Codes:**

| | |
|---|---|
| PRTI_PROC_NOT_CONFIGURED | Process is not configured as network process (Failure) |
| PRTI_FILE_OPEN_ERR | Error opening one of the configuration files (Failure) |
| PRTI_SVC_ID_NDEF | PRT service identifier not valid (Failure) |
| PRTI_FILE_READ_ERR | Error reading configuration file (Failure) |
| PRTI_EMPTY_FILE | Configuration file has no records (Failure) |
| PRTI_AUX_NAM_ERR | Error in **ipc_aux_nam** (Failure) |
| PRTI_ADD_PORT_ERR | Error in **ipc_add_port** (Failure) |
| PRTI_READ_PORT_ERR | Error in **ipc_read_port** (Failure) |
| PRTI_MF_INIT_ERR | Error initializing segment (Failure) |
| PRTI_CRESEG_ERR | Error creating segment (Failure) |
| PRTI_BAD_MSG | Error in segment (Failure) |
| PRTI_BAD_SVC_NAME | Bad service name (no underscore) (Failure) |
| PRTI_XLATE_ERR | Error in **ipc_xlate** getting physical addr (Failure) |
| PRTI_WRITE_PORT_ERR | Error in **ipc_write_port** (Warn/Fail) |

# prt_api_dyn_register_intproc - Register As A Dynamically Interested Process

**Name:** **prt_api_dyn_register_intproc**

**Description:**

This function provides a mechanism for a process to request to register as an interested process to a PRT process. Either this initialization routine must be called by all PRT interested processes prior to calling any other PRT_API interested process utilities or **prt_api_ register_intproc**. The calling routine receives back a status indicating the success or failure of the request. This routine is used when you have *not* configured your process as a service and is *not* included in the **prt_service.dat** file. Otherwise you must use **prt_api_register_intproc**.

**Syntax:**

```
int prt_api_dyn_register_intproc (intproc_event_flag,
                        prt_system, cimp_sys, retstat)

COR_I4      intproc_event_flag;
char        *prt_system;
char        *cimp_sys;
COR_STATUS  *retstat;
```

**Data Structures:**

None

**Input Arguments:**

| | |
|---|---|
| intproc_event_flag | Interested process event flag (required) |
| prt_system | Prefix of PRT data server(s) to connect to. |
| cimp_sys | Name of project or node where prt data server resides. |

**Output Arguments:**

| | |
|---|---|
| *retstat | Pointer to COR_STATUS structure |

**Return Value:**

Either COR_SUCCESS or COR_FAILURE.

If the function returns COR_FAILURE additional error information can be found in the COR_STATUS structure.

**Error Codes:**

| | |
|---|---|
| PRTI_PROC_NOT_CONFIGURED | Process is not configured as network process (Failure) |
| PRTI_FILE_OPEN_ERR | Error opening one of the configuration files (Failure) |
| PRTI_SVC_ID_NDEF | PRT service identifier not valid (Failure) |
| PRTI_FILE_READ_ERR | Error reading configuration file (Failure) |
| PRTI_EMPTY_FILE | Configuration file has no records (Failure) |
| PRTI_AUX_NAM_ERR | Error in **ipc_aux_nam** (Failure) |
| PRTI_ADD_PORT_ERR | Error in **ipc_add_port** (Failure) |
| PRTI_READ_PORT_ERR | Error in **ipc_read_port** (Failure) |

| | |
|---|---|
| PRTI_MF_INIT_ERR | Error initializing segment (Failure) |
| PRTI_CRESEG_ERR | Error creating segment (Failure) |
| PRTI_BAD_MSG | Error in segment (Failure) |
| PRTI_BAD_SVC_NAME | Bad service name (no underscore) (Failure) |
| PRTI_XLATE_ERR | Error in **ipc_xlate** getting physical addr (Failure) |
| PRTI_WRITE_PORT_ERR | Error in **ipc_write_port** (Warn/Fail) |

# prt_api_intproc_region - Receive updates only from specific regions

**Name:** `prt_api_intproc_region`

**Description:**

This function provides a mechanism for a process to request that only specific region data is required. A flag is passed to specify if the region should be added, replaced, or deleted. The calling routine receives back a status indicating the success or failure of the request.

**Syntax:**

```
int prt_api_intproc_region (region_id, action_code, retstat)

char        *region_id;
COR_U4      action_code;
COR_STATUS  *retstat;
```

**Data Structures:**

None

**Input Arguments:**

| | |
|---|---|
| `region_id` | Region that this call applies to |
| `action_code` | Code to specify what action needs to occur. Valid values are:<br>PRT_FILTER_ADD - Add this **`region_id`** to filter list.<br>PRT_FILTER_REPLACE - Replace existing filter **`region_id`** with this one, if one exists. Otherwise add **`region_id`**<br>PRT_FILTER_DELETE - Delete this **`region_id`** from filter list. |

**Output Arguments:**

| | |
|---|---|
| `*retstat` | Pointer to COR_STATUS structure |

**Return Value:**

Either COR_SUCCESS or COR_FAILURE.

If the function returns COR_FAILURE additional error information can be found in the COR_STATUS structure.

**Error Codes:**

| | |
|---|---|
| `IPC_ERR_PARTDEAD` | Partner Dead (Warning) |
| `PRTI_UNKNOWN_INIT_SRC` | MF_INIT segment received from unknown service (Warning) |
| `PRTI_UNEXPECTED_SEG` | Unexpected segment type (Warning) |
| `PRTI_BAD_MSG` | Error in segment (Failure) |
| `PRTI_READ_PORT_ERR` | Error in **`ipc_read_port`** (Failure) |
| `PRTI_EF_CHECK_ERR` | Error checking event flag (Failure) |

# prt_api_intproc_retry_connect - Send INIT Segment to all data servers

**Name:** `prt_api_intproc_retry_connect`

**Description:**

This function provides a mechanism for a process to send an INIT segment to all data servers we are interested in. This routine allows the user to easily reestablish the connection to a data server when the connection has been broken.

**Syntax:**

```
COR_BOOLEAN prt_api_intproc_retry_connect ()
```

**Data Structures:**

None

**Input Arguments:**

None

**Output Arguments:**

None

**Return Value:**

Either TRUE or FALSE.

If the function returns TRUE, then this means that an INIT segment was sent to at least one data server.

**Error Codes:**

None

# prt_api_get_intproc_updates - Get Interested Process Item/Region Tracking Data Updates

**Name:** **prt_api_get_intproc_updates**

**Description:**

This routine provides a mechanism for an application to receive interested process updates from all data servers which have been specified based on the **prt_system** argument in the **prt_api_init** and **prt_api_dyn_register_intproc** calls. The information received by the interested process includes updates for Serialized and Non-serialized Items as well as region-related updates.

There are currently two types of updates that an interested process may receive from PRT:

- An item has been modified or moved - the manifest constant for this update type is PRT_INTPROC_ITEM_UPDATE. The event_code field in the PRT_ITEM_UPDATE_STRUCT indicates the exact nature of the update.
- The items in a region have been reordered - the manifest constant for this update type is PRT_INTPROC_REGION_UPDATE.

With both types of update structure, the action field contains the same value as the **update_type** argument returned to the calling program.

The caller allocates storage for both item update and region update structures. When the call returns, **update_type** will be set to PRT_INTPROC_ITEM_UPDATE if the item update structure has been filled in with item information. If the region update structure has been filled in, **update_type** will be set to PRT_INTPROC_REGION_UPDATE.

If a Data Server process which the interested process is configured to receive interested process updates from terminates (or some other error occurs) the interested process will be notified by receiving a message with **update_type** set to PRT_ERROR_RESP. In this case the service id of the associated Data Collector process will be placed in the user_or_svc_id field of the PRT_ITEM_UPDATE_STRUCT structure.

When **prt_api_get_intproc_updates** is called, it empties the datagram receive buffer of all messages that have arrived since the last call to **prt_api_get_intproc_updates** (or since the process registered as an interested process) via successive reads, until a read is posted and the event flag remains low. The messages which are read are placed into one of two lists, one containing errors messages, and the other containing valid interested process update messages. If there are any messages in the error message list, these are always returned before the real updates.

**Syntax:**

```
int prt_api_get_intproc_updates( update_type, item_info,
                                 reg_info, retstat )

COR_U2                    *update_type;
PRT_ITEM_UPDATE_STRUCT    *item_info;
PRT_REGION_UPDATE_STRUCT  *reg_info;
COR_STATUS                *retstat;
```

**Data Structures:**

```
typedef struct
{
  char att_name[PRT_ATTRIBUTE_LEN + 1] - item attribute name
  char att_value[PRT_ATTRIBUTE_LEN + 1] - item attribute value
} PRT_ITEM_ATT

typedef struct
{
  long         reg_entry_time;
  COR_I2       event_code;
  COR_U2       modify_bitmask;
  char         region_1_id[PRT_REGION_ID_LEN + 1];
  COR_I2       region_1_loc;
  char         region_2_id[PRT_REGION_ID_LEN + 1];
  COR_I2       region_2_loc;
  COR_I1       item_tracking_type;
  char         item_id[PRT_ITEM_ID_LEN + 1];
  char         reference_id[PRT_ITEM_ID_LEN + 1];
  char         item_type_id[PRT_ITEM_TYPE_ID_LEN + 1];
  char         parent_item_id[PRT_ITEM_ID_LEN + 1];
  COR_U4       item_status;
  COR_BOOLEAN  int_hold_active;
  char         int_hold_reason[SC_DESCRIPTION_LEN + 1];
  COR_BOOLEAN  ext_hold_active;
  char         group_id[PRT_REGION_ID_LEN + 1];
  COR_I1       num_atts_valid;
  PRT_ITEM_ATT item_att_list[PRT_NUM_ATTRIBUTES];
  char         comment[SC_DESCRIPTION_LEN + 1];
  char         user_or_svc_id[SERVICE_ID_LEN + 1];
  COR_I1       action;
} PRT_ITEM_UPDATE_STRUCT, *PRT_ITEM_UPDATE_STRUCT_PTR;

typedef struct
{
  COR_I1 event_code;
  char   region_id[PRT_REGION_ID_LEN + 1];
  COR_U4 region_status;
  char   comment[SC_DESCRIPTION_LEN + 1];
  char   user_or_svc_id[SERVICE_ID_LEN + 1];
  COR_I2 location;
  COR_I1 action;
} PRT_REGION_UPDATE_STRUCT, *PRT_REGION_UPDATE_STRUCT_PTR;
```

**Notes:** See the next chapter for the list of macros that can be used with the **`modify_bitmask`** parameter to determine which fields in the PRT_ITEM_UPDATE_STRUCT contain valid data.

The values supplied which identify region location are in the range FIRST (or EXIT), LAST (or ENTRY), and 0, 1, 2...n. '0' means that the region location is not appropriate. 1-n is the region location. FIRST and LAST are first and last to exit, respectively.

In the current implementation of production tracking, **`region_2_id`** and **`region_2_loc`** are not populated in the item update structure. These elements are reserved for GE Fanuc use.

**Input Arguments:**

| | |
|---|---|
| `item_info` | Update information storage location on an item |
| `reg_info` | Update information storage location on a region |

**Output Arguments:**

| | |
|---|---|
| `*update_type` | Interested process update type |
| `item_info` | Updated information on an item |
| `reg_info` | Updated information on a region |
| `*retstat` | Pointer to COR_STATUS structure |
| | **Note:** See the chapter on event codes for a complete listing of the possible event codes for either item or region updated information. |

**Return Value:**

Either COR_SUCCESS, COR_FAILURE or COR_WARNING.

If the function returns COR_WARNING or COR_FAILURE, additional error information can be found in the COR_STATUS structure.

If there are no requests pending, the function returns COR_WARNING with the COR_STATUS err_code set to PRTI_NOTHING_PENDING and **`update_type`** set to PRT_ERROR_RESP.

**Error Codes:**

| | |
|---|---|
| `IPC_ERR_PARTDEAD` | Partner Dead (Warning) |
| `PRTI_UNKNOWN_INIT_SRC` | MF_INIT segment received from unknown service (Warning) |
| `PRTI_INIT_MSG_RCVD` | Init segment received from Data Collector (Warning) |
| `PRTI_UNEXPECTED_SEG` | Unexpected segment type (Warning) |
| `PRTI_NOTHING_PENDING` | No requests pending (Warning) |
| `PRTI_BAD_MSG` | Error in segment (Failure) |
| `PRTI_READ_PORT_ERR` | Error in **`ipc_read_port`** (Failure) |
| `PRTI_EF_CHECK_ERR` | Error checking event flag (Failure) |

# prt_api_term_intproc - Terminate PRT_API Interested Process Interface

**Name:** `prt_api_term_intproc`

**Description:**

This routine is responsible for deallocating all memory allocated by the PRT_API interested process routines and for terminating any services to other application programmer interfaces.

**Syntax:**

```
int prt_api_term_intproc (retstat)

COR_STATUS *retstat;
```

**Data Structures:**

See Static Information Requirements

**Input Arguments:**

None

**Output Arguments:**

| | |
|---|---|
| ***retstat** | Pointer to COR_STATUS structure |

**Return Value:**

None

# prt_api_static_region_data - Copy Static Region Information Into Region Structure

**Name:** **`prt_api_static_region_data`**

**Description:**

This function copies the static part of the region information into the region structure supplied by the application. Note that only region-configured capacity, status and quantity points are returned. If an invalid region is supplied, the region_id field is not populated.

**Syntax:**

```
int prt_api_static_region_data (region_struct, region_id)

PRT_REGION_STRUCT  region_list[list_size];
char               *region_id;
```

**Data Structures:**

```
typedef struct {
  char   region_id[PRT_REGION_ID_LEN + 1] - unique identifier
                                           of tracking region.
  COR_U4 region_status - region status.
  COR_I4 total_items - total number of items in region
  COR_I2 region_cap - number of item carriers in region
  char   stat_ptid[POINT_ID_LEN + 1] - status point identifier
  char   item_qty_ptid[POINT_ID_LEN + 1] - item quantity point id
  COR_I2 total_item_counts - tells how many PRT_ITEM_COUNTS
                             elements are in the array pointed
                             to by type_totals.
  PRT_ITEM_COUNTS *type_totals - pointer to array of
                                 PRT_ITEM_COUNTS structures
                     (array must be deallocated by the user)
} PRT_REGION_STRUCT, *PRT_REGION_STRUCT_PTR;
```

**Input Arguments:**

| | |
|---|---|
| `region_struct` | Pointer to a buffer to receive tracking region data (required) |
| `region_id` | Character string identifying the region of interest. |

**Output Arguments:**

| | |
|---|---|
| `region_struct` | Populated tracking region data |

**Return Value:**

If the region_id field in the **`region_struct`** buffer contains the supplied **`region_id`**, then the configured region quantity in the region_cap field, the status point in the stat_ptid field, and the item quantity point in the item_qty_ptid field are all populated with valid data; otherwise, the region could not be found, and the fields contain invalid data.

# prt_api_prod_start - Initiate Production Start For An Item

**Name:** **`prt_api_prod_start`**

**Description:**

This routine provides a mechanism to initiate production start for an Item in a Tracking Region.

**Syntax:**

```
int prt_api_prod_start ( ext_proc_flag, region_id, item_id,
                         reference_id, item_type_id,
                         parent_item_id, item_status,
                         ext_hold_active, group_id,
                         int_hold_active, int_hold_reason,
                         num_atts_valid, item_att_list,
                         comment, user_or_svc_id, retstat)

COR_BOOLEAN  ext_proc_flag;
char         region_id;[PRT_REGION_ID_LEN + 1]
char         item_id[PRT_ITEM_ID_LEN + 1];
char         reference_id[PRT_ITEM_ID_LEN + 1];
char         item_type_id[PRT_ITEM_TYPE_ID_LEN + 1];
char         parent_item_id[PRT_ITEM_ID_LEN + 1];
COR_U4       item_status;
COR_BOOLEAN  ext_hold_active;
char         group_id[PRT_REGION_ID_LEN + 1];
COR_BOOLEAN  int_hold_active;
char         *int_hold_reason;
COR_I1       num_atts_valid;
PRT_ITEM_ATT *item_att_list;
char         comment[SC_DESCRIPTION_LEN + 1];
char         user_or_svc_id[SERVICE_ID_LEN + 1];
COR_STATUS   *retstat;
```

**Data Structures:**

See Static Information Requirements

**Input Arguments:**

Serialized Items Only

| | |
|---|---|
| `ext_proc_flag` | External process flag; if TRUE, get all information from external process. (required) |
| `item_id` | Unique identifier of a Serialized Item (optional if **`reference_id`** specified) |
| `reference_id` | Secondary identifier of a Serialized Item (optional if **`item_id`** specified) |
| `ext_hold_active` | If TRUE, item held due to external hold (required). |
| `group_id` | Group identifier providing reason for hold (required if **`ext_hold_active`** = TRUE). |

| | |
|---|---|
| num_atts_valid | Number of item attributes valid. This specifies the number of array elements in the item attribute list that contain valid data. If **item_att_list** is a NULL pointer, the function sends a zero to the Data Collector for this argument, regardless of what was actually passed. |
| *item_att_list | Item attribute list. This argument is the pointer to an array of PRT_NUM_ATTRIBUTES elements. If the pointer is NULL, this clearly indicates that no item attributes are being provided along with the item being added. If the pointer is non-NULL, however, it must point to an array of exactly PRT_NUM_ATTRIBUTES elements. The **num_atts_valid** argument indicates how many of these elements contain valid data. |
| Non-Serialized Items Only | |
| parent_item_id | Associated serialized item identifier (optional). |
| Both | |
| region_id | Region where the item is to be added to the queue (required) |
| item_type_id | Item type identifier (required) |
| int_hold_active | If TRUE, item held due to internal hold. (required) |
| int_hold_reason | Comment specifying reason for internal hold (required if **int_hold_active** = TRUE). |
| comment | Comment to be recorded in PRT history log file (optional). |
| user_or_svc_id | User or service identifier. This is used for logging purposes and is optional. If this pointer is NULL, the host process id is used (optional). |

**Output Arguments:**

| | |
|---|---|
| ***retstat** | Pointer to COR_STATUS structure |

**Return Value:**

Either COR_SUCCESS or COR_FAILURE.

If the function returns COR_FAILURE additional error information can be found in the COR_STATUS structure.

**Error Codes:**

| | |
|---|---|
| PRTC_NON_SER_ID | Non-serialized item has item or ref. id (Failure) |
| PRTC_SER_NO_ID | Serialized item missing both item & ref. id (Failure) |
| PRTC_UNKNOWN_REGION | Unknown REGION specified (Failure) |
| PRTC_BAD_REG_LOC_SPEC | Bad region location specified (Failure) |
| PRTC_INS_IN_OCC_NON_NORMAL | Attempt to Insert in occupied COMBINE/DISPERSE region (Failure) |
| PRTC_ITEM_STAMP_TOO_NEW | An affected item has changed since the last display (Failure) |

| | |
|---|---|
| PRTC_ADD_LOC_TOO_BIG | Can not add item to unoccupied location- inserting (Warning) |
| PRTC_ADD_MULTIPLE_SERIAL | Can not have two serialized items at one location (Warning) |
| PRTC_NO_DETAINMENT_CFG | No detainment region configured for (Warning) |
| PRTI_ITEM_TYPE_NULL | Item type identifier NULL (Warning) |
| PRTI_ITEM_TYPE_NDEF | Item type identifier not valid (Warning) |
| PRTI_NS_EXT_PROC_INVALID | **ext_proc_flag** set for non- serialized item (Warning) |
| PRTI_NON_SER_ID | Non-serialized item has item id or reference id (Warning) |
| PRTI_SER_NO_ID | Serialized item missing both item id and reference id (Warning) |
| PRTI_SER_PARENT | Parent ID specified for serialized item (Warning) |
| PRTI_GRP_ID_NULL | Group identifier NULL (Warning) |
| PRTI_GRP_ID_NDEF | Group identifier not valid (Warning) |
| PRTI_EXT_HOLD_NON_SER | External hold specified for non-serialized item (Warning) |
| PRTI_INT_HOLD_NO_REASON | Internal hold specified without reason (Warning) |
| PRTI_NON_SER_ATTS | Non-serialized item has attribute(s) (Warning) |
| PRTI_NUM_ATTS_INVALID | Invalid number of item attributes specified (Warning) |
| PRTI_REG_ID_NULL | Region identifier NULL (Warning) |
| PRTI_REG_ID_NDEF | Region identifier not valid (Warning) |
| PRTI_MF_INIT_ERR | Error initializing segment (Failure) |
| PRTI_CRESEG_ERR | Error creating segment (Failure) |

Plus all error codes returnable by:

- **prt_api_check_route**
- **prt_api_send_cmd**

# prt_api_prod_stop - Enable Production Stop For An Item

**Name:** **`prt_api_prod_stop`**

**Description:**

This API provides a mechanism for removing an Item from the Production Tracking system as the result of its exiting out of a region along a route that has no destination region. The specified Item must currently reside at a source region associated with such a route. If the Item cannot legitimately transition out of the system along such a route, an error is returned to the calling routine.

**Syntax:**

```
int prt_api_prod_stop ( timestamp, region_id, region_loc, item_id,
                        reference_id, item_type_id, comment,
                        user_or_svc_id, retstat )

time_t     timestamp;
char       region_id[PRT_REGION_ID_LEN + 1];
COR_I2     region_loc;
char       item_id[PRT_ITEM_ID_LEN + 1];
char       reference_id[PRT_ITEM_ID_LEN + 1];
char       item_type_id[PRT_ITEM_TYPE_ID_LEN + 1];
char       comment[SC_DESCRIPTION_LEN + 1];
char       user_or_svc_id[SERVICE_ID_LEN + 1];
COR_STATUS *retstat;
```

**Data Structures:**

None

**Input Arguments:**

Serialized Items Only

| | |
|---|---|
| `item_id` | Unique identifier of a serialized item (optional if **`reference_id`** specified) |
| `reference_id` | Secondary identifier of a serialized item (optional if **`item_id`** specified) |

Both Serialized and Non-Serialized Items

| | |
|---|---|
| `timestamp` | The time the item was last moved or modified. The timestamp is used to ensure that, when attempting to transition the specified item out of the production tracking system, the item referred to has not moved or been modified between the time when information was last obtained for that item and the attempt to 'stop' it. (One of the fields in the structure which returns data on an item is the last_mod_time field, which indicates the time at which the item was last moved or modified.)<br><br>If the timestamp argument provided to this function is older than the timestamp associated with the item at the time of the attempted production stop, then the modification is not performed and a warning returned.<br><br>If the timestamp argument passed to this function is zero, the operation is performed without any timestamp checking. |
| `region_id` | Identifier of the Tracking Region where the Item is to be deleted from the queue required) |
| `region_loc` | Region location where the non-serialized item resides (required for non-serialized items, optional for serialized items) |
| `item_type_id` | Item type identifier (required for non-serialized items) |
| `comment` | Comment to be recorded in PRT history log file optional) |
| `user_or_svc_id` | User or service identifier (optional) |

**Output Arguments:**

| | |
|---|---|
| `*retstat` | Pointer to COR_STATUS structure |

**Return Value:**

Either COR_SUCCESS or COR_FAILURE.

If the function returns COR_FAILURE additional error information can be found in the COR_STATUS structure.

**Error Codes:**

| | |
|---|---|
| `PRTC_XREF_NOFIND` | Deleted, but not found in xref table (Warning) |
| `PRTC_ITEM_NOT_FOUND` | Item ID not found (Failure) |
| `PRTC_REF_NOT_FOUND` | Reference ID not found (Failure) |
| `PRTC_BAD_REG_SPEC` | Bad region specified (Failure) |
| `PRTC_BAD_REG_LOC_SPEC` | Bad region location specified (Failure) |
| `PRTC_NO_ITEM_AT_LOC` | Specified item not found specified location (Failure) |
| `PRTC_NO_REG_LOC_SPEC` | Must specify region location to delete non-serialized item (Failure) |

| | |
|---|---|
| PRTC_ITEM_STAMP_TOO_NEW | An affected item has changed since the last display (Failure) |
| PRTI_ITEM_TYPE_NULL | Item type identifier NULL (Warning) |
| PRTI_ITEM_TYPE_NDEF | Item type identifier not valid (Warning) |
| PRTI_NON_SER_ID | Non-serialized item has item id or reference id (Warning) |
| PRTI_SER_NO_ID | Serialized item missing both item id and reference id (Warning) |
| PRTI_REG_ID_NULL | Region identifier NULL (Warning) |
| PRTI_REG_ID_NDEF | Region identifier not valid (Warning) |
| PRTI_INVALID_REG_LOC | Invalid region location (Warning) |
| PRTI_MF_INIT_ERR | Error initializing segment (Failure) |
| PRTI_CRESEG_ERR | Error creating segment (Failure) |

Plus all error codes returnable by:

- **prt_api_send_cmd**
- **prt_api_check_route** (except PRTI_INVALID_ROUTE)

# prt_api_add_item - Add Item To Tracking Region Queue

**Name:** `prt_api_add_item`

**Description:**

This routine provides a mechanism to add an individual Item at the same location where another Item currently resides. Defined constants are provided as a way of indicating that the item is to be added either at the head or at the tail of the queue. The region location can be specified as FIRST (or also as EXIT) to add the item to the first (head) location in the queue, i.e. this item is being added to a location where one or more other items already currently reside (although no more than one serialized item may reside at a region location at any one time). These items will be the next items to transition out of the queue. LAST (and ENTRY) are provided to add the item to the last location in the region (the last location which currently has one or more items residing in it). These items will be the last items to transition out of the region (of all the items currently in the region).

**Syntax:**

```
int prt_api_add_item ( timestamp, ext_proc_flag, region_id,
                       region_loc, item_id, reference_id,
                       item_type_id, parent_item_id,
                       item_status, ext_hold_active,
                       group_id, int_hold_active,
                       int_hold_reason, num_atts_valid,
                       item_att_list, comment, user_or_svc_id,
                       retstat)

time_t       timestamp;
COR_BOOLEAN  ext_proc_flag;
char         region_id[PRT_REGION_ID_LEN + 1];
COR_I2       region_loc;
char         item_id[PRT_ITEM_ID_LEN + 1];
char         reference_id[PRT_ITEM_ID_LEN + 1];
char         item_type_id[PRT_ITEM_TYPE_ID_LEN + 1];
char         parent_item_id[PRT_ITEM_ID_LEN + 1];
COR_U4       item_status;
COR_BOOLEAN  ext_hold_active;
char         group_id[PRT_REGION_ID_LEN + 1];
COR_BOOLEAN  int_hold_active;
char         *int_hold_reason;
COR_I1       num_atts_valid;
PRT_ITEM_ATT *item_att_list;
char         comment[SC_DESCRIPTION_LEN + 1];
char         user_or_svc_id[SERVICE_ID_LEN + 1];
COR_STATUS   *retstat;
```

**Data Structures:**

See Static Information Requirements

**Input Arguments:**

Serialized Items Only

| | |
|---|---|
| `ext_proc_flag` | External process flag; if TRUE, get all information from external process. (required). |
| `item_id` | Unique identifier of a Serialized Item (optional if **`reference_id`** specified) |
| `reference_id` | Secondary identifier of a Serialized Item (optional if **`item_id`** specified) |
| `ext_hold_active` | If TRUE, item held due to external hold. required) |
| `group_id` | Group identifier providing reason for hold (required if **`ext_hold_active`** = TRUE). |
| `num_atts_valid` | Number of item attributes valid. This specifies the number of array elements in the item attribute list that contain valid data. If **`item_att_list`** is a NULL pointer, the function sets this argument to zero regardless of what was actually passed. (required) |
| `*item_att_list` | Item attribute list. This argument is the pointer to an array of PRT_NUM_ATTRIBUTES elements. If the pointer is NULL, this clearly indicates that no item attributes are being provided along with the item being added. If the pointer is non-NULL, however, it must point to an array of exactly PRT_NUM_ATTRIBUTES elements. The **`num_atts_valid`** argument indicates how many of these elements contain valid data. (optional) |

Non-Serialized Items

| | |
|---|---|
| `parent_item_id` | Associated serialized item identifier (optional). |

Both

| | |
|---|---|
| `timestamp` | The time the item was last moved or modified. The timestamp is used to ensure that, when attempting to add an item, the item(s) currently at the location where the new item is being added have not moved or been modified between the time when information was last obtained for that item(s) and the attempt to add a new item at the same location. (One of the fields in the structure which returns data on an item is the last_mod_time field, which indicates the time at which the item was last moved or modified.)<br><br>If the timestamp argument provided to this function is older than the timestamp associated with the item(s) at the location being added to, then the 'add' function is not performed and a warning returned. If the timestamp argument passed to this function is zero, the operation is performed without any timestamp checking. |
| `region_id` | Region where the item is to be added to the queue (required). |

| | |
|---|---|
| region_loc | Specific location in the region where the Item is to be added (required). |
| item_type_id | Item type identifier (required). |
| item_status | Item status (required). |
| int_hold_active | If TRUE, item held due to internal hold. (required) |
| int_hold_reason | Comment specifying reason for internal hold (required if **int_hold_active** = TRUE). |
| comment | Comment to be recorded in PRT history log file. (optional) |
| user_or_svc_id | User or service identifier. This is used for logging purposes and is optional. If this pointer is NULL, the host process id is used. (optional) |

**Output Arguments:**

| | |
|---|---|
| *retstat | Pointer to COR_STATUS structure |

**Return Value:**

Either COR_SUCCESS or COR_FAILURE.

If the function returns COR_FAILURE additional error information can be found in the COR_STATUS structure.

**Error Codes:**

| | |
|---|---|
| PRTC_NON_SER_ID | Non-serialized item has item or ref. id (Failure) |
| PRTC_SER_NO_ID | Serialized item missing both item & ref. id (Failure) |
| PRTC_UNKNOWN_REGION | Unknown REGION specified (Failure) |
| PRTC_BAD_REG_LOC_SPEC | Bad region location specified (Failure) |
| PRTC_ITEM_STAMP_TOO_NEW | An affected item has changed since the last display (Failure) |
| PRTC_ADD_LOC_TOO_BIG | Can not add item to unoccupied location- inserting (Warning) |
| PRTC_ADD_MULTIPLE_SERIAL | Can not have two serialized items at one location (Warning) |
| PRTI_ITEM_TYPE_NULL | Item type identifier NULL (Warning) |
| PRTI_ITEM_TYPE_NDEF | Item type identifier not valid (Warning) |
| PRTI_NS_EXT_PROC_INVALID | **ext_proc_flag** set for non-serialized item (Warning) |
| PRTI_NON_SER_ID | Non-serialized item has item id or reference id (Warning) |
| PRTI_SER_NO_ID | Serialized item missing both item id and reference id (Warning) |
| PRTI_SER_PARENT | Parent ID specified for serialized item (Warning) |
| PRTI_GRP_ID_NULL | Group identifier NULL (Warning) |
| PRTI_GRP_ID_NDEF | Group identifier not valid (Warning) |
| PRTI_EXT_HOLD_NON_SER | External hold specified for non-serialized item (Warning) |
| PRTI_INT_HOLD_NO_REASON | Internal hold specified without reason (Warning) |

| | |
|---|---|
| PRTI_NON_SER_ATTS | Non-serialized item has attribute(s) (Warning) |
| PRTI_NUM_ATTS_INVALID | Invalid number of item attributes specified (Warning) |
| PRTI_REG_ID_NULL | Region identifier NULL (Warning) |
| PRTI_REG_ID_NDEF | Region identifier not valid (Warning) |
| PRTI_INVALID_REG_LOC | Invalid region location (Warning) |
| PRTI_ITEM_TYPE_NULL | Item type identifier NULL (Warning) |
| PRTI_INVALID_DEST | Destination region is invalid for item of this type (Warning) |
| PRTI_MF_INIT_ERR | Error initializing segment (Failure) |
| PRTI_CRESEG_ERR | Error creating segment (Failure) |

Plus all error codes returnable by **prt_api_send_cmd**

# prt_api_insert_item - Insert Item Into Tracking Region Queue

**Name:** **prt_api_insert_item**

**Description:**

This routine provides a mechanism to insert an Item into the Tracking Region queue. If an Item is to be inserted into the queue at a particular location (other than at the head or tail of the queue), the region location of the item to be inserted ahead of is specified, i.e. specifying **region_loc** = 1, is equivalent to specifying **region_loc** = FIRST. Defined constants are provided as a way of indicating that the item is to be inserted either at the head or at the tail of the queue. The region location can be specified as FIRST (or also as EXIT) to insert the item as the first (head) item in the queue, i.e. this item will be the next item to transition out of the queue. LAST (and ENTRY) are provided to insert the item as the last item at the tail of the queue, BEHIND all other items.

**Syntax:**

```
int prt_api_insert_item ( timestamp, ext_proc_flag, region_id,
                          region_loc, item_id, reference_id,
                          item_type_id, parent_item_id,
                          item_status, ext_hold_active,
                          group_id, int_hold_active,
                          int_hold_reason, num_atts_valid,
                          item_att_list, comment,
                          user_or_svc_id, retstat )
time_t       timestamp;
COR_BOOLEAN  ext_proc_flag;
char         region_id[PRT_REGION_ID_LEN + 1];
COR_I2       region_loc;
char         item_id[PRT_ITEM_ID_LEN + 1];
char         reference_id[PRT_ITEM_ID_LEN + 1];
char         item_type_id[PRT_ITEM_TYPE_ID_LEN + 1];
char         parent_item_id[PRT_ITEM_ID_LEN + 1];
COR_U4       item_status;
COR_BOOLEAN  ext_hold_active;
char         group_id[PRT_REGION_ID_LEN + 1];
COR_BOOLEAN  int_hold_active;
char         *int_hold_reason;
COR_I1       num_atts_valid;
PRT_ITEM_ATT *item_att_list;
char         comment[SC_DESCRIPTION_LEN + 1];
char         user_or_svc_id[SERVICE_ID_LEN + 1];
COR_STATUS   *retstat;
```

**Data Structures:**

See Static Information Requirements

**Input Arguments:**

Serialized Items Only

| | |
|---|---|
| ext_proc_flag | External process flag; if TRUE, get all information from external process. (required) |
| item_id | Unique identifier of a Serialized Item (optional if **reference_id** is specified) |
| reference_id | Secondary identifier of a Serialized Item (optional if **item_id** is specified) |

| | |
|---|---|
| ext_hold_active | If TRUE, item held due to external hold. (required) |
| `group_id` | Group identifier providing reason for hold (required if **`ext_hold_active`** = TRUE). |
| `num_atts_valid` | Number of item attributes valid; this specifies the number of array elements in the item attribute list that contain valid data. If **`item_att_list`** is a NULL pointer, the function sends a zero to the Data Collector for this argument, regardless of what was actually passed. (required) |
| `item_att_list` | Item attribute list. This argument is the pointer to an array of PRT_NUM_ATTRIBUTES elements. If the pointer is NULL, this clearly indicates that no item attributes are being provide for the item being added. If the pointer is non-NULL, however, it must point to an array of exactly PRT_NUM_ATTRIBUTES elements. The **`num_atts_valid`** argument indicates how many of these elements contain valid data. (optional) |
| Non-Serialized Items Only | |
| `parent_item_id` | Associated serialized item identifier (optional). |
| Both | |
| `timestamp` | The time the item was last moved or modified. The timestamp is used to ensure that, when attempting to insert an item, the item(s) currently at the location where the new item is to be inserted in front of have not moved or been modified between the time when information was last obtained for that item(s) and the attempt to insert the new item in front of it them). (One of the fields in the structure which returns data on an item is the last_mod_time field, which indicates the time at which the item was last moved or modified.)<br><br>If the timestamp argument provided to this function is older than the timestamp associated with the item(s) at the location where the new item is to be inserted in front of, then the 'insert' function is not performed and a warning returned. If the timestamp argument passed to this function is zero, the operation is performed without any timestamp checking. |
| `region_id` | Region where the item is to be added to the queue (required). |
| `region_loc` | Specific location in the region where the Item is to be added (required argument). Specified as FIRST (EXIT), LAST (ENTRY), or a number from 1 to n, where n is the number of items in the queue. (required) |
| `item_type_id` | Item type identifier (required). |
| `item_status` | Item status (required). |
| `int_hold_active` | If TRUE, item held due to internal hold. (required) |

| | |
|---|---|
| `int_hold_reason` | Comment specifying reason for internal hold (required if **`int_hold_active`** = TRUE). |
| `comment` | Comment to be recorded in PRT history log file. (optional) |
| `user_or_svc_id` | User or service identifier. This is used for logging purposes and is optional. If this pointer is NULL, the host process id is used. (optional) |

**Output Arguments:**

| | |
|---|---|
| *****`retstat` | Pointer to COR_STATUS structure |

**Return Value:**

Either COR_SUCCESS or COR_FAILURE.

If the function returns COR_FAILURE additional error information can be found in the COR_STATUS structure.

**Error Codes:**

| | |
|---|---|
| `PRTC_NON_SER_ID` | Non-serialized item has item or ref. id (Failure) |
| `PRTC_SER_NO_ID` | Serialized item missing both item & ref. id (Failure) |
| `PRTC_UNKNOWN_REGION` | Unknown REGION specified (Failure) |
| `PRTC_BAD_REG_LOC_SPEC` | Bad region location specified (Failure) |
| `PRTC_INS_IN_OCC_NON_NORMAL` | Attempt to Insert in occupied COMBINE/DISPERSE region (Failure) |
| `PRTC_ITEM_STAMP_TOO_NEW` | An affected item has changed since the last display (Failure) |
| `PRTC_ADD_LOC_TOO_BIG` | Can not add item to unoccupied location- inserting (Warning) |
| `PRTC_ADD_MULTIPLE_SERIAL` | Can not have two serialized items at one location (Warning) |
| `PRTC_NO_DETAINMENT_CFG` | No detainment region configured for (Warning) |
| `PRTI_ITEM_TYPE_NULL` | Item type identifier NULL (Warning) |
| `PRTI_ITEM_TYPE_NDEF` | Item type identifier not valid (Warning) |
| `PRTI_NS_EXT_PROC_INVALID` | **`ext_proc_flag`** set for non-serialized item (Warning) |
| `PRTI_NON_SER_ID` | Non-serialized item has item id or reference id (Warning) |
| `PRTI_SER_NO_ID` | Serialized item missing both item id and reference id (Warning) |
| `PRTI_SER_PARENT` | Parent ID specified for serialized item (Warning) |
| `PRTI_GRP_ID_NULL` | Group identifier NULL (Warning) |
| `PRTI_GRP_ID_NDEF` | Group identifier not valid (Warning) |
| `PRTI_EXT_HOLD_NON_SER` | External hold specified for non-serialized item (Warning) |
| `PRTI_INT_HOLD_NO_REASON` | Internal hold specified without reason (Warning) |
| `PRTI_NON_SER_ATTS` | Non-serialized item has attribute(s) (Warning) |

| | |
|---|---|
| PRTI_NUM_ATTS_INVALID | Invalid number of item attributes specified (Warning) |
| PRTI_REG_ID_NULL | Region identifier NULL (Warning) |
| PRTI_REG_ID_NDEF | Region identifier not valid (Warning) |
| PRTI_INVALID_REG_LOC | Invalid region location (Warning) |
| PRTI_MF_INIT_ERR | Error initializing segment (Failure) |
| PRTI_CRESEG_ERR | Error creating segment (Failure) |

Plus all error codes returnable by **prt_api_send_cmd**

# prt_api_modify_region - Modify Tracking Region Data

**Name:** **prt_api_modify_region**

**Description:**

This routine provides a mechanism to modify Tracking Region Data; specifically, the region status.

**Syntax:**

```
int prt_api_modify_region ( status_bitmask, region_id,
                            region_status, comment,
                            user_or_svc_id, retstat )

COR_U4     status_bitmask;
char       region_id;[PRT_REGION_ID_LEN + 1]
COR_U4     region_status;
char       comment[SC_DESCRIPTION_LEN + 1];
char       user_or_svc_id[SERVICE_ID_LEN + 1];
COR_STATUS *retstat;
```

**Data Structures:**

See Static Information Requirements

**Input Arguments:**

| | |
|---|---|
| `status_bitmask` | Indicates the bit positions in the **`region_status`** that are to be modified. (required) |
| `region_id` | Region identifier (required) |
| `region_status` | Tracking Region status code (e.g. IN-LOCKED, OUT-LOCKED, FULL, OUT-OF-SEQ). (optional) |
| `comment` | Comment to be recorded in PRT history log file (optional). |
| `user_or_svc_id` | User or service identifier. This is used for logging purposes and is optional. If this pointer is NULL, the host process id is used. (optional) |

**Output Arguments:**

| | |
|---|---|
| `*retstat` | Pointer to COR_STATUS structure |

**Return Value:**

Either COR_SUCCESS or COR_FAILURE.

If the function returns COR_FAILURE additional error information can be found in the COR_STATUS structure.

**Error Codes:**

| | |
|---|---|
| `PRTC_BAD_REG_SPEC` | Bad region specified (Warning) |
| `PRTI_REG_ID_NULL` | Region identifier NULL (Warning) |
| `PRTI_REG_ID_NDEF` | Region identifier not valid (Warning) |
| `PRTI_MF_INIT_ERR` | Error initializing segment (Failure) |
| `PRTI_CRESEG_ERR` | Error creating segment (Failure) |

Plus all error codes returnable by **`prt_api_send_cmd`**

# prt_api_move_item - Move Item To Location In Tracking Region

**Name:** **prt_api_move_item**

**Description:**

This routine provides a mechanism for moving an Item from the Tracking Region where it currently resides to a new Tracking Region. It also permits the movement of an individual Item from one location to another within the same Tracking Region. The semantics of this call specify that if neither the **item_id, reference_id**, or **item_type_id** is specified, all items are moved from the specified source to the specified destination region. If neither the **item_id or reference_id** is specified, but the **item_type_id** is, a single item of the specified type is moved from the source to the destination region. If we're moving a serialized item, the source location need not be given. The specification of the destination region location argument is the same as that for the **prt_api_add_item** function if the **insert_flag** passed in is FALSE, and is the same as that for the **prt_api_insert_item** function if the **insert_flag** passed in is TRUE.

**Syntax:**

```
int prt_api_move_item ( src_reg_id, dest_reg_id, src_reg_loc,
                        dest_reg_loc, insert_flag, item_id,
                        reference_id, item_type_id, comment,
                        user_or_svc_id, retstat)

char        src_reg_id[PRT_REGION_ID_LEN + 1];
char        dest_reg_id[PRT_REGION_ID_LEN + 1];
COR_I2      src_reg_loc;
COR_I2      dest_reg_loc;
COR_BOOLEAN insert_flag;
char        item_id[PRT_ITEM_ID_LEN + 1];
char        reference_id[PRT_ITEM_ID_LEN + 1];
char        item_type_id[PRT_ITEM_TYPE_ID_LEN + 1];
char        comment[SC_DESCRIPTION_LEN + 1];
char        user_or_svc_id[SERVICE_ID_LEN + 1];
COR_STATUS  *retstat;
```

**Data Structures:**

See Static Information Requirements

**Input Arguments:**

Serialized Items Only

| | |
|---|---|
| item_id | Unique identifier of a Serialized Item (optional if **reference_id** is specified). |
| reference_id | Secondary identifier of a Serialized Item (optional if **item_id** is specified). |

Both Serialized and Non-Serialized Items

| | |
|---|---|
| src_reg_id | Source region id; region the item is moving out of (required). |
| dest_reg_id | Destination region id; region the item is moving into (required). |

| | |
|---|---|
| src_reg_loc | Source region location; sequence number of the item in the source tracking region. Specified as FIRST (EXIT), LAST (ENTRY), or a number from 1 to n, where n is the number of items in the queue. required for non-serialized items, optional for serialized items). |
| dest_reg_loc | Destination region location; sequence number of the item in the destination tracking region. Specified as FIRST (EXIT), LAST (ENTRY), or a number from 1 to n, where n is the number of items in the queue. (required). |
| insert_flag | If TRUE, the moved item is to be inserted in the Destination Tracking Region queue before the item located at the specified destination region location, unless the manifest constant LAST (ENTRY) is specified, in which case the item is inserted after behind) the last item currently present in the region. (required). |
| item_type_id | Item type identifier (required for non-serialized items, optional for serialized items). |
| comment | Comment to be recorded in PRT history log file (optional). |
| user_or_svc_id | User or service identifier. This is used for logging purposes and is optional. If this pointer is NULL, the host process id is used. (optional) |

**Output Arguments:**

| | |
|---|---|
| *retstat | Pointer to COR_STATUS structure |

**Return Value:**

Either COR_SUCCESS or COR_FAILURE.

If the function returns COR_FAILURE additional error information can be found in the COR_STATUS structure.

**Error Codes:**

| | |
|---|---|
| PRTC_NON_SER_ID | Non-serialized item has item or ref. id (Failure) |
| PRTC_SER_NO_ID | Serialized item missing both item & ref. id (Failure) |
| PRTC_UNKNOWN_REGION | Unknown REGION specified (Failure) |
| PRTC_BAD_REG_LOC_SPEC | Bad region location specified (Failure) |
| PRTC_ITEM_STAMP_TOO_NEW | An affected item has changed since the last display (Failure) |
| PRTC_ADD_LOC_TOO_BIG | Can not add item to unoccupied location- inserting (Warning) |
| PRTC_ADD_MULTIPLE_SERIAL | Can not have two serialized items at one location (Warning) |
| PRTC_REGION_NOT_LOCKED | Lockable region %s not locked; cannot perform operation (Warning) |
| PRTI_ITEM_TYPE_NULL | Item type identifier NULL (Warning) |
| PRTI_ITEM_TYPE_NDEF | Item type identifier not valid (Warning) |

| | |
|---|---|
| PRTI_NON_SER_ID | Non-serialized item has item id or reference id (Warning) |
| PRTI_SER_NO_ID | Serialized item missing both item id and reference id (Warning) |
| PRTI_REG_ID_NULL | Region identifier NULL (Warning) |
| PRTI_REG_ID_NDEF | Region identifier not valid (Warning) |
| PRTI_INVALID_REG_LOC | Invalid region location (Warning) |
| PRTI_ITEM_TYPE_NULL | Item type identifier NULL (Warning) |
| PRTI_INVALID_DEST | Destination region is invalid for item of this type (Warning) |
| PRTI_MF_INIT_ERR | Error initializing segment (Failure) |
| PRTI_CRESEG_ERR | Error creating segment (Failure) |

Plus all error codes returnable by **prt_api_send_cmd**

# prt_api_move_item_to_detain - Move Item To Detainment Region

**Name:** **prt_api_move_item_to_detain**

**Description:**

This routine provides a mechanism for moving an Item from the Tracking Region where it currently resides to the detainment region of that region. This function checks that a detainment region exists for the source region and then calls **prt_api_move_item** to actually complete the work.

**Syntax:**

```
int prt_api_move_item_to_detain ( src_reg_id, src_reg_loc,
                        item_id, reference_id, item_type_id,
                        comment, user_or_svc_id, retstat)

char        src_reg_id[PRT_REGION_ID_LEN + 1];
COR_I2      src_reg_loc;
char        item_id[PRT_ITEM_ID_LEN + 1];
char        reference_id[PRT_ITEM_ID_LEN + 1];
char        item_type_id[PRT_ITEM_TYPE_ID_LEN + 1];
char        comment[SC_DESCRIPTION_LEN + 1];
char        user_or_svc_id[SERVICE_ID_LEN + 1];
COR_STATUS  *retstat;
```

**Data Structures:**

See Static Information Requirements

**Input Arguments:**

Serialized Items Only

| | |
|---|---|
| item_id | Unique identifier of a Serialized Item (optional if **reference_id** is specified). |
| reference_id | Secondary identifier of a Serialized Item (optional if **item_id** is specified). |

Both Serialized and Non-Serialized Items

| | |
|---|---|
| src_reg_id | Source region id; region the item is moving out of (required). |
| src_reg_loc | Source region location; sequence number of the item in the source tracking region. Specified as FIRST (EXIT), LAST (ENTRY), or a number from 1 to n, where n is the number of items in the queue. required for non-serialized items, optional for serialized items). |
| item_type_id | Item type identifier (required for non-serialized items, optional for serialized items). |
| comment | Comment to be recorded in PRT history log file (optional). |
| user_or_svc_id | User or service identifier. This is used for logging purposes and is optional. If this pointer is NULL, the host process id is used. (optional) |

**Output Arguments:**

| | |
|---|---|
| `*retstat` | Pointer to COR_STATUS structure |

**Return Value:**

Either COR_SUCCESS or COR_FAILURE.

If the function returns COR_FAILURE additional error information can be found in the COR_STATUS structure.

**Error Codes:**

| | |
|---|---|
| `PRTC_NON_SER_ID` | Non-serialized item has item or ref. id (Failure) |
| `PRTC_SER_NO_ID` | Serialized item missing both item & ref. id (Failure) |
| `PRTC_UNKNOWN_REGION` | Unknown REGION specified (Failure) |
| `PRTC_BAD_REG_LOC_SPEC` | Bad region location specified (Failure) |
| `PRTC_ITEM_STAMP_TOO_NEW` | An affected item has changed since the last display (Failure) |
| `PRTC_ADD_LOC_TOO_BIG` | Can not add item to unoccupied location- inserting (Warning) |
| `PRTC_ADD_MULTIPLE_SERIAL` | Can not have two serialized items at one location (Warning) |
| `PRTC_REGION_NOT_LOCKED` | Lockable region %s not locked; cannot perform operation (Warning) |
| `PRTI_ITEM_TYPE_NULL` | Item type identifier NULL (Warning) |
| `PRTI_ITEM_TYPE_NDEF` | Item type identifier not valid (Warning) |
| `PRTI_NON_SER_ID` | Non-serialized item has item id or reference id (Warning) |
| `PRTI_SER_NO_ID` | Serialized item missing both item id and reference id (Warning) |
| `PRTI_REG_ID_NULL` | Region identifier NULL (Warning) |
| `PRTI_REG_ID_NDEF` | Region identifier not valid (Warning) |
| `PRTI_INVALID_REG_LOC` | Invalid region location (Warning) |
| `PRTI_ITEM_TYPE_NULL` | Item type identifier NULL (Warning) |
| `PRTI_INVALID_DEST` | Destination region is invalid for item of this type (Warning) |
| `PRTI_MF_INIT_ERR` | Error initializing segment (Failure) |
| `PRTI_CRESEG_ERR` | Error creating segment (Failure) |

Plus all error codes returnable by **`prt_api_send_cmd`**

# prt_api_check_route - Validate Tracking Route

**Name:** `prt_api_check_route`

**Description:**

This routine provides a mechanism for validating that an ITEM of a specific type is configured to move from a Source Tracking Region to a Destination Tracking Region. Route validation is performed locally by the API from configuration information that was read into the API's internal schema during initialization. If the function returns COR_SUCCESS, the route is valid. If some other status is returned, the err_code field of the COR_STATUS structure will contain additional information. If a valid route does exist between the source and destination regions specified (according to the configuration data) but the item type specified is not allowed to transition along this route, PRTI_INVALID_ROUTE is returned in the err_code field. If there are no routes configured between the source and destination regions specified, the err_code field is set to PRTI_NO_SUCH_ROUTE.

**Syntax:**

```
int prt_api_check_route ( src_region_id, dest_region_id,
                          item_type_id, retstat )

char       src_region_id[PRT_REGION_ID_LEN + 1];
char       dest_region_id;[PRT_REGION_ID_LEN + 1]
char       item_type_id[PRT_ITEM_TYPE_ID_LEN + 1];
COR_STATUS *retstat;
```

**Data Structures:**

See Static Information Requirements

**Input Arguments:**

| | |
|---|---|
| `src_region_id` | Source region id; the starting region of the route (required). |
| `dest_region_id` | Destination region id; the destination region of the route (required). |
| `item_type_id` | Item type identifier (required). |

**Output Arguments:**

| | |
|---|---|
| `*retstat` | Pointer to COR_STATUS structure |

**Return Value:**

Either COR_SUCCESS or COR_FAILURE.

If the function returns COR_FAILURE additional error information can be found in the COR_STATUS structure.

**Error Codes:**

| | |
|---|---|
| `PRTI_REG_ID_NULL` | Region identifier NULL (Warning) |
| `PRTI_REG_ID_NDEF` | Region identifier not valid (Warning) |
| `PRTI_ITEM_TYPE_NULL` | Item type identifier NULL (Warning) |
| `PRTI_ITEM_TYPE_NDEF` | Item type identifier not valid (Warning) |
| `PRTI_NO_SRC_OR_DEST` | Route has no source and no destination region (Warning) |
| `PRTI_INVALID_ROUTE` | Invalid route for item type (Warning) |
| `PRTI_NO_SUCH_ROUTE` | No route exists between src/dest regions (Warning) |

## prt_api_advance_model - Advance Tracking Model

**Name:** **`prt_api_advance_model`**

**Description:**

This routine provides a mechanism for advancing the next Item Carrier to leave a Source Tracking Region to a Destination Tracking Region.

**Note:** This call moves *all* items at the FIRST (EXIT) location in the source region to the LAST (ENTRY) location in the destination region.

**Syntax:**

```
int prt_api_advance_model ( src_reg_id, dest_reg_id, comment,
                            user_or_svc_id, retstat )

char       src_reg_id[PRT_REGION_ID_LEN + 1];
char       dest_reg_id[PRT_REGION_ID_LEN + 1];
char       comment[SC_DESCRIPTION_LEN + 1];
char       user_or_svc_id[SERVICE_ID_LEN + 1];
COR_STATUS *retstat;
```

**Data Structures:**

See Static Information Requirements

**Input Arguments:**

| | |
|---|---|
| `src_reg_id` | Source region id; region the item is moving out of (required). |
| `dest_reg_id` | Destination region id; region the item is moving into (required). |
| `comment` | Comment to be recorded in PRT history log file (optional). |
| `user_or_svc_id` | User or service identifier. This is used for logging purposes and is optional. If this pointer is NULL, the host process id is used. (optional) |

**Output Arguments:**

| | |
|---|---|
| *retstat | Pointer to COR_STATUS structure |

**Return Value:**

Either COR_SUCCESS or COR_FAILURE.

If the function returns COR_FAILURE additional error information can be found in the COR_STATUS structure.

**Error Codes:**

All error codes returnable by **`prt_api_move_item`**

# prt_api_region_setpoints - Enable/Disable Tracking Region Setpoints

**Name:** **prt_api_region_setpoints**

**Description:**

This routine provides a mechanism for enabling and disabling the Item transition setpoints configured in the PRT Tracking Route records. These points are set by PRT when an Item transitions into the configured destination region.

**Syntax:**

```
int prt_api_region_setpoints ( region_id, enable_setpt_flag,
                               comment, user_or_svc_id, retstat )

char        region_id[PRT_REGION_ID_LEN + 1];
COR_BOOLEAN enable_setpt_flag;
char        comment[SC_DESCRIPTION_LEN + 1];
char        user_or_svc_id[SERVICE_ID_LEN + 1];
COR_STATUS  *retstat;
```

**Data Structures:**

See Static Information Requirements

**Input Arguments:**

| | |
|---|---|
| region_id | Tracking Region identifier (required) |
| enable_setpt_flag | If TRUE, enable the region item transition setpoint. If FALSE, disable it. (required) |
| comment | Comment to be recorded in PRT history log file (optional). |
| user_or_svc_id | User or service identifier. This is used for logging purposes and is optional. If this pointer is NULL, the host process id is used. (optional) |

**Output Arguments:**

| | |
|---|---|
| ***retstat** | Pointer to COR_STATUS structure |

**Return Value:**

Either COR_SUCCESS or COR_FAILURE.

If the function returns COR_FAILURE additional error information can be found in the COR_STATUS structure.

**Error Codes:**

| | |
|---|---|
| PRTC_BAD_REG_SPEC | Bad region specified (Warning) |
| PRTI_REG_ID_NULL | Region identifier NULL (Warning) |
| PRTI_REG_ID_NDEF | Region identifier not valid (Warning) |
| PRTI_MF_INIT_ERR | Error initializing segment (Failure) |
| PRTI_CRESEG_ERR | Error creating segment (Failure) |

Plus all error codes returnable by **prt_api_send_cmd**

# prt_api_reorder_region - Reorder locations within Tracking Region

**Name:** **`prt_api_reorder_region`**

**Description:**

This routine provides a mechanism to completely reorder the locations within a tracking region. This has the effect of moving all the items at each location in the region to another (possibly the same) location in the region.

**Syntax:**

```
int prt_api_reorder_region ( region_id, reorder_list,
                             num_locations, comment,
                             user_or_svc_id, retstat )

char       region_id[PRT_REGION_ID_LEN + 1];
COR_I2     reorder_list[num_locations];
COR_I2     num_locations;
char       comment[SC_DESCRIPTION_LEN + 1];
char       user_or_svc_id[SERVICE_ID_LEN + 1];
COR_STATUS *retstat;
```

**Data Structures:**

See Static Information Requirements

**Input Arguments:**

| | |
|---|---|
| `region_id` | Tracking Region identifier (required) |
| `reorder_list` | Pointer to an array consisting of region locations. These location numbers (1 to **`num_locations`**), specify the new region location for items residing in their current location, e.g. the locations the items are to be moved to. The **`reorder_list`** array index implicitly identifies the current location number, i.e. the first element in the **`reorder_list`** array (**`reorder_list[0])`** refers to the items that currently reside in region location 1. The value of **`reorder_list[0]`** specifies the region location where the items currently in region location 1 are to be moved to. This array must include elements for each location currently occupied in the region. |
| `num_locations` | The number of locations in the region which currently are occupied. This number must match what the Data Collector thinks are the number of locations currently occupied or else the Data Collector will return an error (**`err_code`** = PRTC_REORDER_MISMATCH) and the operation will not be performed. Failure to allocate the **`reorder_list`** array with **`num_locations`** elements is bad. |
| `comment` | Comment to be recorded in PRT history log file (optional). |

| | |
|---|---|
| `user_or_svc_id` | User or service identifier. This is used for logging purposes and is optional. If this pointer is NULL, the host process id is used. (optional) |

**Output Arguments:**

| | |
|---|---|
| `*retstat` | `Pointer to COR_STATUS structure` |

**Return Value:**

Either COR_SUCCESS or COR_FAILURE.

If the function returns COR_FAILURE additional error information can be found in the COR_STATUS structure.

**Error Codes:**

| | |
|---|---|
| `PRTC_BAD_REG_SPEC` | Bad region specified (Warning) |
| `PRTC_REORDER_MISMATCH` | Num. of locs in request different from num. of locs in region (Warning) |
| `PRTI_REG_ID_NULL` | Region identifier NULL (Warning) |
| `PRTI_REG_ID_NDEF` | Region identifier not valid (Warning) |
| `PRTI_REORD_NUM_TOO_BIG` | Element # bigger than list size (Warning) |
| `PRTI_MF_INIT_ERR` | Error initializing segment (Failure) |
| `PRTI_CRESEG_ERR` | Error creating segment (Failure) |

Plus all error codes returnable by **`prt_api_send_cmd`**

## prt_api_next_svc - Return each successive configured PRT Service

**Name:** `prt_api_next_svc`

**Description:**

The first time this routine is called, the **`svc_rec`** should be the address of a NULL pointer. The schema record associated with the first configured PRT service will be returned in the **`svc_rec`** and the service id copied into the **`prt_svc_id`** string. Each subsequent call will use the **`svc_rec`** pointer to get the next service (by search order, as configured in the PRT_SERVICE file) and return a pointer to this record in **`svc_rec`** (as well as copying the **`prt_svc_id`**).

**Syntax:**

```
int prt_api_next_svc ( svc_rec, prt_svc_id )

struct prt_service **svc_rec;
char   prt_svc_id[SERVICE_ID_LEN + 1];
```

**Data Structures:**

See Static Information Requirements

**Input Arguments:**

| | |
|---|---|
| `svc_rec` | Pointer to a pointer to a PRT_SERVICE record (record in the PRT_API internal schema which contains information on configured PRT services. |

**Output Arguments:**

| | |
|---|---|
| `svc_rec` | Pointer to a pointer to a PRT_SERVICE record (record in the PRT_API internal schema which contains information on configured PRT services. Each successive call returns a pointer to a new PRT_SERVICE record. |
| `prt_svc_id` | PRT Service identifier |

**Return Value:**

Either COR_SUCCESS or COR_WARNING.

COR_SUCCESS is returned as long as the next service has been found. When there are no more services to be returned (the entire schema has been run through), COR_WARNING is returned.

# prt_api_get_item_types - Return a list of all configured item types

**Name:** **`prt_api_get_item_types`**

**Description:**

This routine returns a list of all the configured item types in the PRT system. It also returns a counter for the number of items found.

**Syntax:**

```
int prt_api_get_item_types ( item_type_ids, total, retstat )

char          **item_type_ids;
COR_I4        *total;
COR_STATUS    *retstat;
```

**Data Structures:**

None

**Input Arguments:**

None

**Output Arguments:**

| | |
|---|---|
| `item_type_ids` | Pointer to a pointer to list of item ids. |
| `total` | Pointer to an integer with the total number of item ids in the **`item_type_ids`** list. |

**Return Value:**

Either COR_SUCCESS or COR_WARNING.

COR_SUCCESS is returned as long as the list exists. COR_WARNING is returned if no list exists.

# prt_api_get_item_tracking_type - Return the tracking type for a specific item

**Name:** `prt_api_get_item_tracking_type`

**Description:**

This function returns the tracking type for a specific item id. A value of 1 is returned if the item id is a serialized item otherwise the value 2 is returned.

**Syntax:**

```
int prt_api_get_item_tracking_type ( item_type_id,
                                     tracking_type, retstat )

char            *item_type_id;
COR_I1          *tracking_type;
COR_STATUS      *retstat;
```

**Data Structures:**

None

**Input Arguments:**

| | |
|---|---|
| `item_type_id` | Pointer to a null terminated string containing a valid item type id. |

**Output Arguments:**

| | |
|---|---|
| `tracking_type` | Pointer to a 1 byte integer specifying the tracking type. A value of 1 is a serialized item, a value of 2 is a non-serialized item. |
| `retstat` | Pointer to COR_STATUS structure. |

**Return Value:**

Either COR_SUCCESS or COR_WARNING.

COR_SUCCESS is returned if the item id has been found. COR_FAILURE is returned if the item id can not be found in any tracking region.

# prti_get_RCMconnection - Connect to Remote Connection Manager

**Name:** **`prti_get_RCMconnection`**

**Description:**

In order to connect to the data server on a remote node, this function must first make sure the remote node is active. Once we know it is active and we have initialized with RCM, RCM will handle the connections later.

**Syntax:**

```
int prti_get_RCMconnection ( cimp_sys, retstat )

char            *cimp_sys;
COR_STATUS      *retstat;
```

**Data Structures:**

None

**Input Arguments:**

| | |
|---|---|
| `cimp_sys` | Pointer to a null terminated string containing the project or node name of the remote system to connect to. |

**Output Arguments:**

| | |
|---|---|
| `retstat` | Pointer to COR_STATUS structure |

**Return Value:**

Either COR_SUCCESS or COR_FAILURE

COR_SUCCESS is returned if the remote system is active. COR_FAILURE is returned if we cannot connect to the remote system.

# prti_term_RCMconnection - Disconnect from Remote Connection Manager

**Name:** **`prti_term_RCMconnection`**

**Description:**

This function breaks the connection to the remote connection manager that was made by **`prti_get_RCMconnection`**.

**Syntax:**

```
int prti_term_RCMconnection ( retstat )

COR_STATUS      *retstat;
```

**Data Structures:**

None

**Input Arguments:**

None

**Output Arguments:**

| | |
|---|---|
| `retstat` | Pointer to COR_STATUS structure |

**Return Value:**

Either COR_SUCCESS or COR_FAILURE

COR_SUCCESS is returned if the remote system is disconnected. COR_FAILURE is returned if we cannot disconnect from the remote system.

# Process Resynchronization

The use of the **cor_dialect_check** function provides a convenient means for the user application to synchronize/resynchronize with the appropriate PRT processes. When an INIT_SEG is received by the application, and invoking the **cor_dialect_check** function returns the PRT subsystem string, the user has the following options:

- If the application is an Interested Process, the **prt_api_register_intproc** function should be called;
- If the application is an API client, the **prt_api_init** function should be called.
- If the application is both Interested Process and API client, both functions should be called.

These routines provide the user with a consistent means for synchronizing with PRT processes when the application program is initializing, and resynchronizing with PRT processes after they have been stopped and restarted.

In order to use the **cor_dialect_check** function, the **cor_dialect.h** header file must be included in your source code file. You must manage your primary port, and call **cor_dialect_check** with the buffer returned from the **ipc_read** on the primary port. The application must also include **%BSM_ROOT%\api\lib\corutil.lib** in the link command file. The calling sequence is as follows:

**Syntax:**

```
void cor_dialect_check ( input_buffer, port_index, result_str )

IPCDG *input_buffer;
int   port_index;
char  result_str[COR_DIALECT_RETURN_STRING+1];
```

**Data Structures:**

None.

**Input Arguments:**

| | |
|---|---|
| input_buffer | Buffer returned from **ipc_read** on primary port |
| port_index | Output port index |

**Output Arguments:**

| | |
|---|---|
| **result**_str | Returned identifying string. See **cor_dialect.h** for valid strings returned. |

**Return Value:**

None.

# PRT API Macros

## About PRT Macros

This chapter includes macros for clearing, setting and testing the various bits of the **modify_bitmask** argument used by the **prt_api_modify_item** and **prt_api_get_intproc_updates** functions.

The **SET** macro sets the bit that corresponds to the actual data field in the item information structure. The **CLR** macro clears the bit. The **IS** macro returns a non-zero value if the bit is set and zero if the bit is clear.

| | |
|---|---|
| **SET_ITEM_ID_MOD_BIT(x)** | |
| **CLR_ITEM_ID_MOD_BIT(x)** | **item_id** |
| **IS_ITEM_ID_MOD(x)** | |
| | |
| **SET_REFERENCE_ID_MOD_BIT(x)** | |
| **CLR_REFERENCE_ID_MOD_BIT(x)** | **reference_id** |
| **IS_REFERENCE_ID_MOD(x)** | |
| | |
| **SET_ITEM_TYPE_BIT(x)** | |
| **CLR_ITEM_TYPE_BIT(x)** | **item_type_id** |
| **IS_ITEM_TYPE(x)** | |
| | |
| **SET_PARENT_ITEM_BIT(x)** | |
| **CLR_PARENT_ITEM_BIT(x)** | **parent_item_id** |
| **IS_PARENT_ITEM(x)** | |
| | |
| **SET_ITEM_STATUS_BIT(x)** | |
| **CLR_ITEM_STATUS_BIT(x)** | **item_status** |
| **IS_ITEM_STATUS(x)** | |

```
SET_INT_HOLD_BIT(x)
CLR_INT_HOLD_BIT(x)             int_hold_active
IS_INT_HOLD(x)                  If int_hold_active = TRUE,
                  int_hold_reason also contains valid data

SET_EXT_HOLD_BIT(x)
CLR_EXT_HOLD_BIT(x)             ext_hold_active
IS_EXT_HOLD(x)                  If ext_hold_active = TRUE,
                        group_id also contains valid data

SET_ATTS_VALID_BIT(x)
CLR_ATTS_VALID_BIT(x)       not needed when using the
IS_ATTS_VALID(x)            'prt_api_modify_item_att' function
```

# Appendix A - PRT API Error Codes

## Error codes Returned by Production Tracking

The following error codes are returned by the Production Tracking Application Interface:

| Number | Defined Constant | Description |
|---|---|---|
| 11401 | PRTI_FILE_READ_ERR | Error reading configuration file: |
| 11402 | PRTI_FILE_OPEN_ERR | Error opening one of the configuration files |
| 11403 | PRTI_SVC_DEF_MISSING | Undefined Service in file: <filename> |
| 11404 | PRTI_REG_DEF_MISSING | Undefined Region in file: <filename> |
| 11405 | PRTI_GRP_DEF_MISSING | Undefined Group in file: <filename> |
| 11406 | PRTI_XLATE_ERR | Error in **`ipc_xlate`** getting physical addr of: <address> |
| 11407 | PRTI_AUX_NAM_ERR | **`Error in ipc_aux_nam`** |
| 11408 | PRTI_ADD_PORT_ERR | **`Error in ipc_add_port`** |
| 11409 | PRTI_READ_PORT_ERR | **`Error in ipc_read_port`** |
| 11410 | PRTI_WRITE_PORT_ERR | **`Error in ipc_write_port`** |
| 11411 | PRTI_MF_INIT_ERR | Error initializing segment |
| 11412 | PRTI_CRESEG_ERR | Error creating segment |
| 11413 | PRTI_BAD_MSG | Error in segment |
| 11414 | PRTI_NOT_STATUS_SEG | Expected STATUS_SEG as response, rcvd: |
| 11415 | PRTI_EXTRA_SEGS | Ignored extra segments after STATUS_SEG or END_SEG |
| 11416 | PRTI_UNEXPECTED_SEG | Unexpected segment type: svc: <filename> |
| 11417 | PRTI_UNEXPECTED_SEG_TYPE | Unexpected segment type from PRT_DS: |
| 11418 | PRTI_ITEM_TYPE_NDEF | Item type identifier not valid: |
| 11419 | PRTI_ITEM_TYPE_NULL | Item type identifier NULL |
| 11420 | PRTI_NUM_ATTS_INVALID | Invalid number of item attributes specified |
| 11421 | PRTI_REG_ID_NDEF | Region identifier not valid: |
| 11422 | PRTI_REG_ID_NULL | Region identifier NULL |
| 11423 | PRTI_GRP_ID_NDEF | Group identifier not valid: |
| 11424 | PRTI_GRP_ID_NULL | Group identifier NULL |
| 11425 | PRTI_SVC_ID_NDEF | PRT service identifier not valid: |
| 11426 | PRTI_SVC_ID_NULL | PRT service identifier NULL |

| | | |
|---|---|---|
| 11427 | PRTI_NS_EXT_PROC_INVALID | **ext_proc_flag** set for non-serialized item |
| 11428 | PRTI_SER_NO_ID | Serialized item missing both item id and reference id |
| 11429 | PRTI_NON_SER_ID | Non-serialized item has item id or reference id |
| 11430 | PRTI_INVALID_TRK_TYPE | Invalid Tracking Type (not SERIALIZED/NON_SERIALIZED) |
| 11431 | PRTI_INVALID_REG_LOC | Invalid region location |
| 11432 | PRTI_EXT_HOLD_NO_GRP | External hold specified without Group ID |
| 11433 | PRTI_EXT_HOLD_NON_SER | External hold specified for non-serialized item |
| 11434 | PRTI_INT_HOLD_NO_REASON | Internal hold specified without reason |
| 11435 | PRTI_REG_NOT_IN_GRP | Region not in group specified |
| 11436 | PRTI_REG_NOT_IN_SVC | Region not managed by service specified |
| 11437 | PRTI_GRP_NOT_IN_SVC | Group ID inconsistent with service specified |
| 11438 | PRTI_NO_DEL_SUCCESSES | No deletes successful to any PRT services |
| 11439 | PRTI_INVALID_DEST | Destination region is invalid for item of this type |
| 11440 | PRTI_NO_SUCH_ROUTE | No route exists between src: <region> dest: <region> |
| 11441 | PRTI_INVALID_ROUTE | Invalid route (%.15s->%.15s) for item type <item type> |
| 11442 | PRTI_SER_PARENT | Parent ID specified for serialized item |
| 11443 | PRTI_EXT_HOLD_NO_REASON | Group hold specified without reason |
| 11444 | PRTI_ROUTE_NDEF | Non-exist. route in PRT_INV_TYPE, src: <region> dest: <region> |
| 11445 | PRTI_INV_EXPLICIT_ROUTE | Can't deny explicit item-route, src: <region> dest: <region> |
| 11446 | PRTI_NO_TYPE_FOR_ROUTE | Denied type not on route, src: <region> dest: <region> type: <item type> |
| 11447 | PRTI_NOTHING_PENDING | No requests pending |
| 11448 | PRTI_CFG_NO_ROUTE | Route has no source and no destination region |
| 11449 | PRTI_EF_CHECK_ERR | Error checking event flag |
| 11450 | PRTI_ITEM_NOT_FOUND | Item not found |
| 11451 | PRTI_ITEM_NOT_FOUND_INC_SRCH | Item not found: some service(s) down |
| 11452 | PRTI_TOO_MANY_REQ | Total number in list less than number requested |
| 11453 | PRTI_UNKNOWN_INIT_SRC | MF_INIT segment received from unknown service |
| 11454 | PRTI_NO_SRC_OR_DEST | Route has no source and no destination region |
| 11455 | PRTI_NO_SVC_IMPLIED | No service specified (via **svc_id/group_id/region_id**) |
| 11456 | PRTI_BAD_SERVER_RESPONSE | PRT Data Server returned bad/error response |
| 11457 | PRTI_NO_MORE_ITEMS | No more items available |
| 11458 | PRTI_BAD_POINTER | Pointer to data structure is NULL |
| 11459 | PRTI_UNKNOWN_RESPONDER | Response received from unknown service |
| 11460 | PRTI_EMPTY_FILE | Configuration file has no records: |
| 11461 | PRTI_BAD_SVC_NAME | Bad service name (no underscore): |
| 11462 | PRTI_PARTNER_DEAD | Partner dead |
| 11463 | PRTI_DG_BUF_FULL | Datagram buffer full |
| 11464 | PRTI_NO_SEGS_RCVD | No segments received into datagram buffer |
| 11465 | PRTI_API_CANT_RCV | PRT Application not initialized to receive asynch messages |
| 11466 | PRTI_INIT_MSG_RCVD | Init segment received from Data Collector |
| 11467 | PRTI_PROC_NOT_CONFIGURED | Process is not configured as network process |
| 11468 | PRTI_NOT_ALL_SVCS_RESPONDED | Not all services responded |

| | | |
|---|---|---|
| 11469 | PRTI_NON_SER_ATTS | Non-serialized item has attribute(s) |
| 11470 | PRTI_CANT_MOD_ITEM_REF_IDS | Can't modify both item & ref.ids |
| 11471 | PRTI_CANT_MOD_ITEM_WOUT_REF | Must specify ref. id to modify item id |
| 11472 | PRTI_REGION_NOT_IN_GROUP | Region <region> does not belong to any group |
| 11473 | PRTI_GROUP_HAS_NO_REGIONS | Group <group> does not 'own' any regions |
| 11474 | PRTI_REORD_NUM_TOO_BIG | element <element> bigger than |
| 11475 | PRTI_GRP_SPANS_SVCS | Can't put region %s in group %s (wrong service) |
| 11476 | PRTI_SVC_NOMATCH | No configured Services for the PRT System: |
| 11477 | PRTI_REGSVC_NOMATCH | No configured Regions for the PRT System: |
| 11478 | PRTI_BAD_ALLOC | Error allocating memory! |
| 11479 | PRTI_NULL_ITEMTYPE_LIST | Bad item type list. |

# Appendix B - Event Codes for Interested Process Updates

## About Event Codes

When **`prt_api_get_intproc_updates`** returns updated information, one of the fields is the event code. This event code, which is a field in either the PRT_ITEM_UPDATE_STRUCT or PRT_REGION_UPDATE_STRUCT, indicates the exact nature of the update.

This chapter presents a listing of all codes that can be received in interested process updates, the occurrences that cause them, and how the **`intproc`** update structure is filled when they occur.

**Note: `Modify_bitmask`** is set to 0 for all codes except IMOD.

### Item-Related Event Codes

IMOD

Automatic item association when adding an item:

PARENT_ITEM_BIT set in **`modify_bitmask`**

**`region_1_id`** contains the region id where the modify occurred

When modifying item:

**`modify_bitmask`** passed from API call

**`region_1_id`** contains the region id where the modify occurred

When item moves and **`transition_type_indicator`** is ASSOCIATE_REF_TO_ITEM or ASSOCIATE_TYPE_TO_ITEM:

**`modify_bitmask`** set to REFERENCE_ID_MOD_BIT or ITEM_TYPE_BIT respectively

**`region_1_id`** and **`region_1_loc`** contain the destination location

ISTT

When adding item via API and **prod_start_flag** is set:

**region_1_id** and **region_1_loc** contain the location of the event

When moving and src region is NULL:

**region_1_id** and **region_1_loc** contain the destination location

IAUX

When serialized item is added to region and xref table indicates that the item was already somewhere within the view of that service, the item already there is deleted.

**region_1_id** and **region_1_loc** contain the location of the item being deleted

When item is being deleted in source region due to a point transition.

**region_1_id** and **region_1_loc** contain the location of the item being deleted

IAUN

When item is being added to destination region due to a point transition.

**region_1_id** and **region_1_loc** contain the destination of the added item

IADD

When item is added via API (or user interface). If item is not production start and is added (not inserted).

**region_1_id** and **region_1_loc** contain the location of the event

IINS

When item is added via API (or user interface). If item is not production start and is inserted (not added).

**region_1_id** and **region_1_loc** contain the location of the event

IMVX

When item is being deleted in source region due to being moved via API (or user interface).

**region_1_id** and **region_1_loc** contain the location of the item being deleted

IMVN

When item is being added to destination region due to being moved via API (or user interface).

**region_1_id** and **region_1_loc** contain the destination of the added item

IADX

When item is being deleted in source region due to being advanced via API (or user interface).

**region_1_id** and **region_1_loc** contain the location of item being deleted

IINS

When item is being added to destination region due to being advanced via API (or user interface).

**region_1_id** and **region_1_loc** contain the destination of added item

IDEL

When item deleted via API (or user interface) and item is being deleted (not scrapped and not production-stopped).

**region_1_id** and **region_1_loc** contain the location of the item being deleted

ISCR

When item deleted via API (or user interface) and item is being scrapped (not deleted and not production-stopped).

**region_1_id** and **region_1_loc** contain the location of the scrapped item

When an item becomes delayed and the region is configured to scrap (rather than detain) delayed items, a message is generated for the item being scrapped.

**region_1_id** and **region_1_loc** contain the location of the scrapped item

IMRA - Aut Repair Move

When a combine or disperse region has a serialized item in it and another serialized item is moved into the region, the item is automatically moved to a detainment region and two messages generated, one for the item being deleted and one for the item being added.

**region_1_id** and **region_1_loc** contain the region from which the item is being deleted

**region_1_id and region_1_loc** contain the destination of detainment region to which the item is being added

When an item becomes delayed and the region is configured to detain (rather than scrap) delayed items, two messages are generated for the item being detained, one for the item being deleted from the combine/disperse region and one for the item being added to the detainment region.

**region_1_id** and **region_1_loc** contain the location of item being deleted

**region_1_id** and **region_1_loc** contain the destination of detainment region to which the item is being added

IMRM - Man Repair Move

When an item is moved via an API call (or user interface) to its detainment region, two messages are generated, one for the item being deleted from the source region and one for the item being added to the detainment region.

**region_1_id** and **region_1_loc** contain the location of the item being deleted

**region_1_id** and **region_1_loc** contain the destination of detainment region to which the item is being added

ISTP - Production Stop

Production stop generated via API call.

**region_1_id** and **region_1_loc** contain the location of the item being stopped

IDRF

This item has been deleted because room was needed in the region

**region_1_id** and **region_1_loc** contain the location of the item being deleted

## Region-related Interested Process Updates

RORD

Generated when a region is reordered via API call or user interface.

**region_1_id** contains the region where the reorder occurred

**modify_bitmask** set to zero

# Appendix C - PRT Rules and Assumptions

## General Rules

A *Tracking Region* is delimited by the Region's defined Entry *Transition Indicator* and the Region's *Exit Transition Indicator*. A Region Entry/Exit Indicator consists of an automatic Item identifier, an automatic Item Type detector, a limit switch, a User Interface transaction or an Application Interface message.

If an Item enters a Region, and that Item occupies a location exceeding the capacity of the Region, configuration data determines if an Alarm is generated.

If an Item enters a *Combine Items Tracking Region*, and that Item's presence in the Region exceeds the capacity of a location in the Region, configuration data determines if an Alarm is generated.

A *Sequential Tracking Region*, monitoring the relative sequence of the Items in the region by their unique Item identifiers, must have:

- An automatic Item identifier *or* a User transaction (specifying Item ID or Reference ID), *or*
- An Application Interface message (specifying Item ID or Reference ID) as a Region Entry Transition Indicator, *or*
- Be preceded by a Sequential Tracking Region also monitoring the relative sequence of the Items in the Region by their unique Item identifiers.

A *Sequential Tracking Region* monitoring the relative sequence of the Items in the Region by their Item type, must have:

- An Item type detector, or
- A User transaction (specifying Item type), or
- An Application Interface message (specifying Item type) as a Region Entry Transition Indicator, or
- Be preceded by a Sequential Tracking Region also monitoring the relative sequence of the Items in the region by their Item Type.

A *Pool Tracking Region* maintains the Items in the tracking queue in the order the Items entered the region.

A *Pool Tracking Region*, monitoring the Items in the Region by their unique Item identifiers, must have:

- An automatic Item Identifier, or
- A User transaction (specifying Item ID or Reference ID), or
- An Application Interface message (specifying Item ID or Reference ID) as a Region Entry Indicator, or
- Be preceded by a Sequential Tracking Region also monitoring the relative sequence of the Items in the region by their unique Item or Reference Identifiers.

A *Pool Tracking Region* monitoring the Items in the Region by their Item Type, must have:

- An Item Type detector, or
- A User transaction (specifying Item Type), or
- An Application Interface message (specifying Item Type) as a Region entry Indicator, or
- Be preceded by a Sequential Tracking Region also monitoring the relative sequence of the Items in the region by their Item Type.

An *Automatic Item Identifier* is equivalent to a User transaction or an Application Interface message, in that it provides PRT with positive unique identification of the Item.

If more than one Entry Transition Indicator feeds a Tracking Region, Items are stored in the region in the order in which the Entry Transition Indicators are received by PRT.

A region Transition Indicator reports the movement of an Item between two Tracking Regions. Each Transition Indicator defines unique routes that exist between two regions. A Transition Indicator is therefore the Exit Transition Indicator for the Source Tracking Region and the Entry Transition Indicator for the Destination Tracking Region. The same Transition Indicator may specify a valid route between two regions for more than one Item Type.

In a Tracking Route, if the Item Type is specified as a blank field or the wildcard character, all Types in the region are assumed to be allowed to travel this route between the Source and Destination Tracking Regions. The Item Type may also contain a mix of text and wildcard characters. The wildcard character is an asterisk (*).

When a region Entry Transition Indicator is received by PRT, PRT updates its tracking data regardless of what tripped the indicator. PRT cannot distinguish between valid Item Transition Indications it receives and accidental Item Transition Indications it receives (e.g. person walks in front of limit switch, PRT receives the Transition Indication and updates the system to reflect Item movement).

Items tracked by PRT must always reside in a Tracking Region.

Production start must always occur in a Tracking Region with an Entry Transition Indicator that is ***not*** the Exit Transition Indicator for any other Tracking Region.

Item movement is directed from the Source Tracking Region to the Destination Tracking Region and cannot reverse direction along a single route.

A Tracking Region may have one and only one Detainment Region.

Multiple regions may utilize the same Detainment Region.

Configuration data determines when an Item is placed in a Detainment Region.

A Tracking Region must be a member of at least one Tracking Region Group. All Tracking Regions in a Tracking Group must be configured for the same PRT process Service ID.

# Combine Items Tracking Region Rules

A *Combine Items Tracking Region* is a region in which multiple Items enter the region through separate region Entry Transition Indicators and reside physically on the same Item Carrier in that region. Multiple Items may also enter through one Entry Transition Indicator by PRT receiving multiple notifications from the Transition Indicator. The Items themselves are in no way required to be associated.

- The Items merely reside on the same physical carrier, they are not required to be associated in any way.
- Multiple region Entry Transition Indications are allowed.
- A single Exit Transition Indication is required from the region.
- The region must be configured to have a region capacity of one.
- Positive Item Type detection is required for each Item entering the region.
- It is not required that all Items arrive at the region before the carrier exits the region. The Items that arrived at the carrier before a transition out of the region is detected are the Items that moved together with that carrier.

A Combine Items Tracking Region is a region in which greater than one Entry Transition Indication is received before an Exit Transition Indication is received.

# Disperse Items Tracking Region Rules

A *Disperse Items Tracking Region* is a region in which a single Item Carrier entered the region carrying multiple Items, but multiple Item Carriers (or equivalent) will exit the region carrying one or more of the Items which resided on the entry Item Carrier.

- A single region Entry Transition Indication is required.
- Multiple region Exit Transition Indications are allowed.
- The region must be configured to have a region capacity of one.
- If a region Exit Transition Indicator has blanks specified for the type of Item that may exit that point all Items are tracked through that exit.
- If a region Exit Transition Indicator has configured a specific Item Type for that exit, only Items of that type are dispersed from the group of Items.
- Positive Item Type identification is required for individually dispersed Items.
- If any of the items in a disperse region is serialized, it must be moved before the other items are dispersed from the region.

If the source region is not a disperse region, the exit transition will move all items at the source location. Otherwise, only the specified item is moved.

PRT determines what Item to move to a Destination Region by using these two rules: If positive identification of the Item (either ID or type) is made available to PRT through information provided by the Transition Indicator, the Item or Item Type identified is moved from the Source Region to the Destination Region. If positive identification is not provided to PRT from a Transition Indicator (e.g. limit switch), the information PRT maintained about the Item(s) in the Source Region will be used.

A Tracking Region designated to execute automatic Item Associations must be a Combine Items Tracking Region.

Item Associations can be specified at run-time via Application Interface messages. The message will provide the child Item Type and the region ID where that Item is the next Item to exit the region and the Item ID of parent Item (Serialized Item) the child (Serialized or Non-serialized Item) is to be associated with.

The association of a Non-serialized Item to a Serialized parent Item can become corrupted or lost in the three ways specified below. PRT has no way of re-associating the Items when an association is lost except through a manual user transaction.

- If a sequence error is detected for a Non-serialized Item in a Tracking Region where the Item has an associated parent Item.
- If a Non-serialized Item crosses a PRT Service boundary and the PRT process for the first service is unable to provide the parent Item data for the Non-serialized Item.
- A Non-serialized Item has an associated parent Item and the Non-serialized Item is placed on an Item Carrier with another Item of the same Item Type.

The concept of Item Hold Specifications apply ***only*** to Serialized Items.

PRT requires positive identification of an Item at the entry to the first region configured for a PRT process (i.e. NODE). Positive identification consists of the Item ID or Reference ID for Serialized Items an the Item Type for Non-serialized Items.

Serialized Items may not be child members of an association.

Sequential Tracking Regions of size one do not detect sequence errors.

A full complement of the PRT configuration data resides on each Node.

When a Destination Region has an Item enter the region, PRT will ask the Source Region about the Item that just transitioned. If the Source Region is a region managed by another PRT process, that process will be requested for information on that Item. If the Source Region has no information on the Item, PRT will request the information from an external process. These requests will be continuously made each time the Item moves, until a response is available. The external process is defined to PRT through configuration data. The configuration of an external process is optional. PRT will only request an external process to provide information on serialized items.

The Tracking Route for a Production Start Tracking Region can specify a limit switch as the Transition Indicator if the Item Type is explicitly defined for the route.

The Tracking Route for a Production Start must be configured to provide the Item ID if the item is to be Serialized.

If a limit switch is used to transition an item and the source region is empty, the limit switch is ignored.

If multiple Items reside at a single Tracking Region Location, one and only one of the Items may be a Serialized Item.

Out of sequence errors will apply to Serialized Items only. Unexpected Item errors will apply to Non-serialized Items.

In PRT Service configuration, the External Process and External Hold Process cannot be the same process.

If an Item identifier or Item type identifier is set and the Item cannot be found, an Item with that ID, reference ID or type (as appropriate) is created.

Processing of Item Identifier points works in the following manner: The first character of the point value is checked when the point value is received. If the character is not '$', the point is expected to contain the configured identifier, and that Item is searched for and moved. If the first character *is* a '$', the rest of the value is checked against the reserved words. If the rest of the value is found to be a reserved word, the corresponding action is taken; otherwise, the rest of the value is taken as an Item Type ID and processing continues as if an Item Type detector was activated.

Currently, the only reserved word is TMOUT, and the action taken is to ignore the point.

# Shifting Region Tracking

*Shifting* regions support empty locations intermixed with occupied locations. As items are transitioned into a *shifting* region, all items currently in the region are shifted by one location, and the item at the *head* location is transitioned to the *Shifting Transition Region*. Items are always transitioned to the *end* location of the region, not the last occupied location.

## Shifting Region Tracking Configuration

To configure a shifting region, the corresponding record in the PRT_REGION file should have :

- **`tracking_type_id`**: SHIFTING
- **`tran_region_id:`** **`region_id`** of the transition region of shifting region.

When a new shifting region is configured, on the subsequent startup PRT will create blank items automatically for all the locations. These blank items are only placeholders; they can not be deleted, added, or moved by the user.

While configuring item types, do not configure any **`item_type_id`** as, which is a GE Fanuc reserved word for **`item_type_id`**s of blank items for the Import/Export utilities.

# Shifting Region Operations

The following sections detail the functionality of shifting regions.

## Adding an item into a Shifting Region

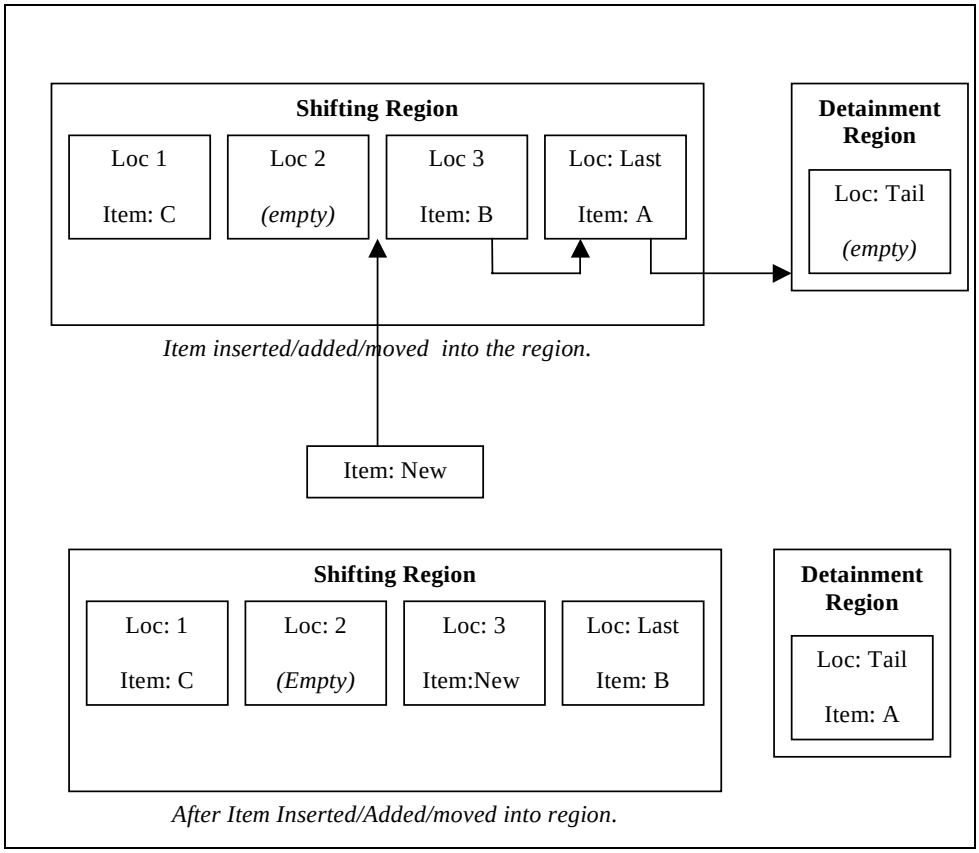
When a new item is added to a shifting region and its location is other than the end location, then the last item in the region is moved to the configured detainment region (if not configured, it is deleted). All the items from the added location to the end location are shifted one location towards the end location. *If the last item was a blank item, then it gets dropped instead of moving into the detainment region. If the configured detainment region is a shifting region, then the item at the last location is not moved to detainment region, but is deleted. A configured detainment region should never be a shifting region.*

However, if a new item is added at the end location, then the head item (first location) in the region is moved to the configured transition region (if not configured, it is deleted) and all items are shifted one location towards the head location.

## Inserting an Item into a Shifting Region

When a new item is inserted into a shifting region and the insert location is other than the end location, then the last item in the region is moved to the detainment region (if configured, otherwise it is deleted) and all the items from the inserted location to the end location are shifted one location towards the end location. *If the last item was a blank item, then it gets dropped instead of moving into the detainment region.*

However, if a new item is inserted at the end location, then the head item (first location) in the region is moved to the configured transition region (if not configured, it is deleted) and all items are shifted one location towards the head location.

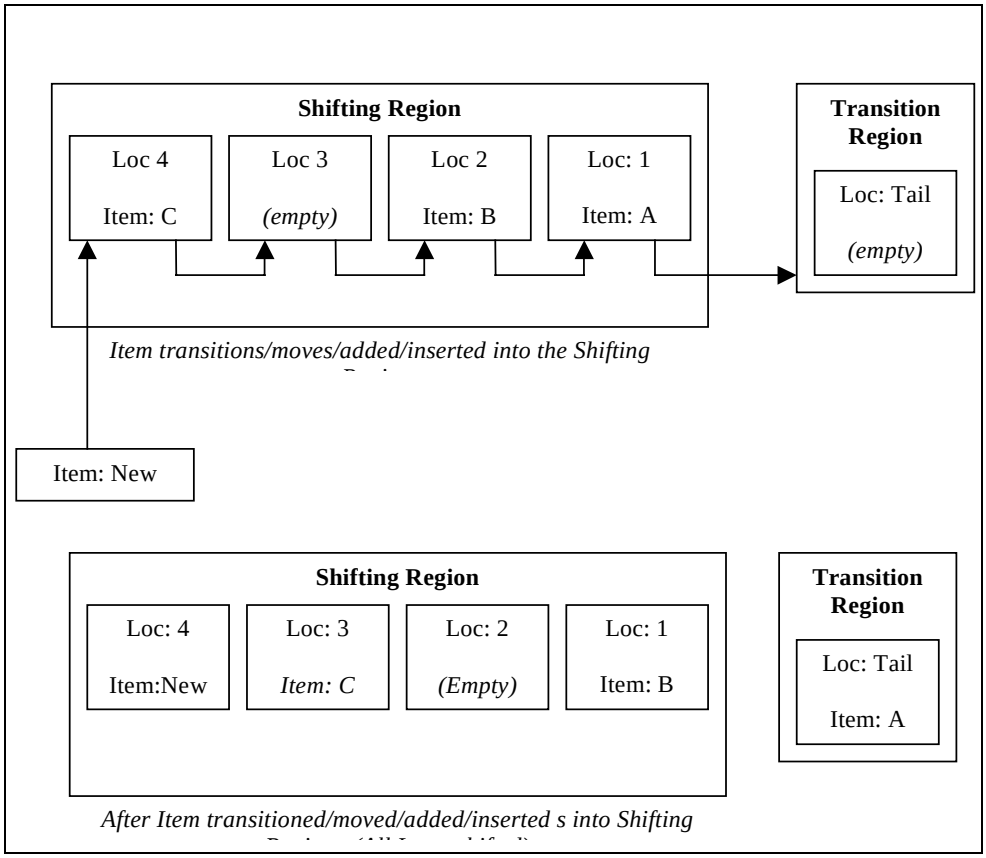


## Moving an Item into a Shifting Region

When a new item is moved into a shifting region and its new location is other than the end location , then the last item in the region is moved to the configured detainment region(if not configured, it is deleted) and all the items from the added location to the end location are shifted one location towards the end location. *If the last item was a blank item, then it gets dropped instead of moving into the detainment region.*

However, if a new item is moved to (transitioned to) the end location then the head item (first location) in the region is moved to the configured transition region (if configured, otherwise it is deleted) and all items are shifted one location towards head.

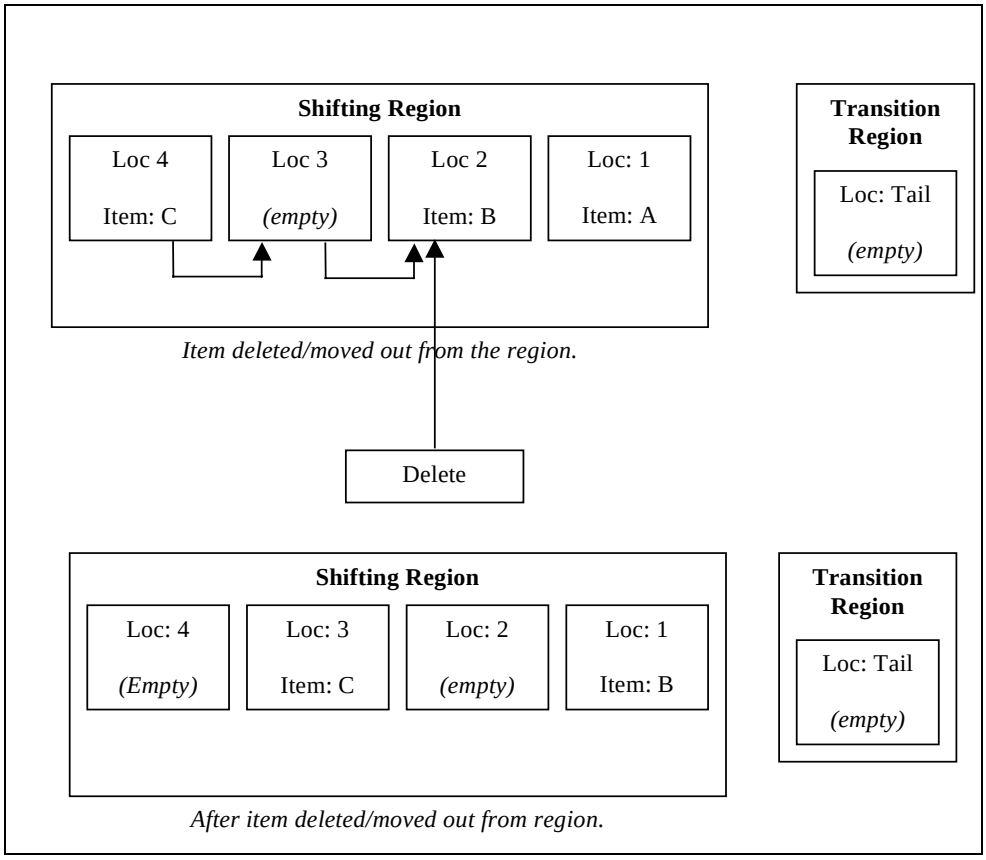
*A move by the user is not allowed for a blank item.*

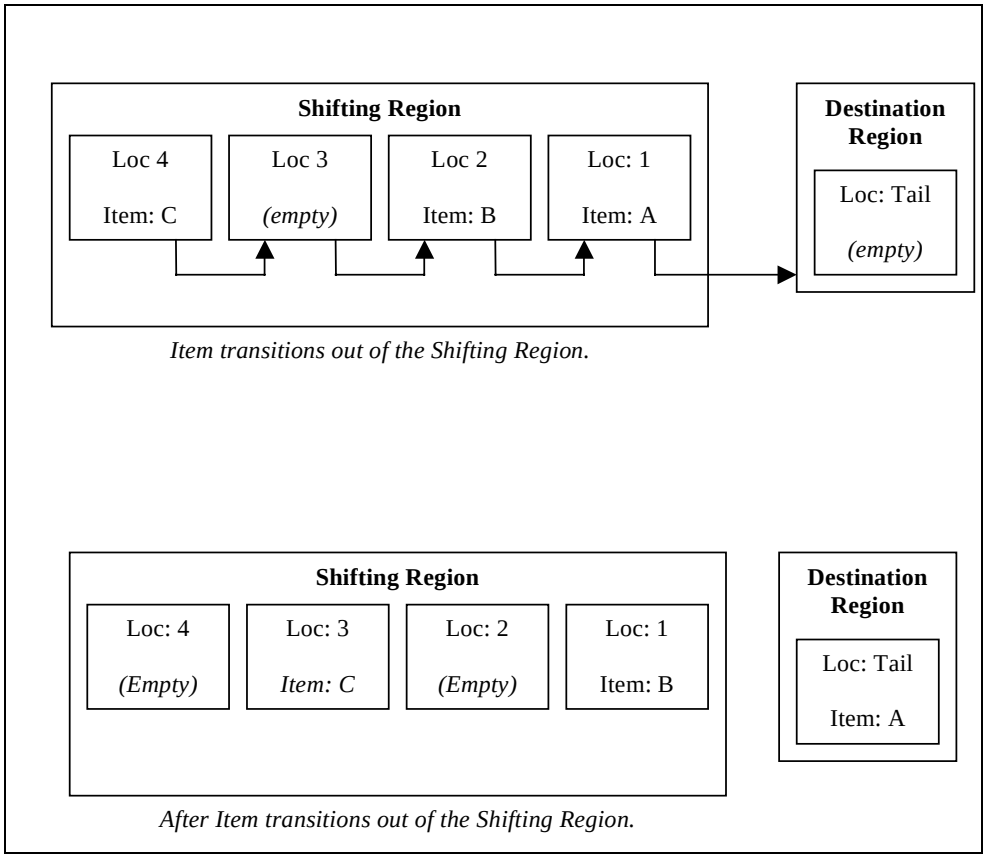


## Deleting or Moving an Item from a Shifting Region

If an item is deleted or moved out from a shifting region, all items from that location to end location are shifted by one location towards the head location and a blank item is added at the end location.

*A Delete by the user is not allowed for a blank item.*






*Item transitions out of the Shifting Region.*

*After Item transitions out of the Shifting Region.*

## Modifying an Item from a Shifting Region

Blank items can not be modified by the user  However, data can be corrected by editing an export file and importing the corrected data. The export file contains one record for each blank location in the shifting region, designated with a reserved item_type_id of **$$GEF@BLK@ITM$$**. No two records having this item_type_id can have the same location number even if the location can contain multiple items.

## Routing

A decision can be processed (for selection/elimination) from a number of decisions based on criteria which involves evaluation (e.g., compare) of the attributes of the last item in the destination region. If the destination region is a shifting region and the last location contains a blank item, then the decision will be dropped from the possible decision list (i.e. this decision will not be processed).

# Shifting Region Process Behavior

The following section describes other process behavior regarding shifting regions.

## PRT

PRT never returns blank items when a request for an item list is made. It will return only the valid items, and the RegionLoc field in the item structure will contain the actual location of that item.

## GLB_VERIFY

GLB_VERIFY checks that the number of occupied locations of a shifting region is equal to the region capacity.

## PRT_UI

PRT_UI will display a blank row for each blank location in a shifting region. Items at these locations can not be modified, deleted, or moved by user.

## PRT_EXPORT/ PRT_IMPORT

While exporting a shifting region, prt_export generates one record for every blank location (even if the location can contain multiple items). This record contains the location number and the item_type_id as **`$$GEF@BLK@ITM$$`**. On import of these records with the reserved **`item_type_ids , prt_import`** generates blank item(s) at those locations.

## PRT_GRD

If the prt_att_pt file is configured to display n to n+m items in a region and if any of the locations in that range is blank, than no data is displayed for that location.

If the prt_att_pt file is configured to display the last n items in a shifting region, then the last n valid items will be displayed. However, if the number of valid items is less than n, then those items will be displayed and the remaining locations will display blanks.

# Appendix D - Using the Global Verifier

## About the Global Verifier

There is a utility available to validate the integrity of the global sections for a given Production Tracking service. It resides in **%BSM_ROOT%\exe** and is called **PRT_GLB_VFY.EXE**. The utility requires three inputs from the user:

- The service name of the PRT data collector that manages the global sections of interest.
- A value that indicates the type of verification. Enter **1** to analyze the global sections, or enter **2** to attempt repairs.
- The name of a file in which to deposit the output.

The program is run from the command line as a utility. It requires that the corresponding PRT data collector and data server be terminated to prevent erroneous messages due to the global section changing during the verification. Note also that this program maps onto the global sections, which means that it must be run on the node where the global section backing files exist.

# Validations and Repairs

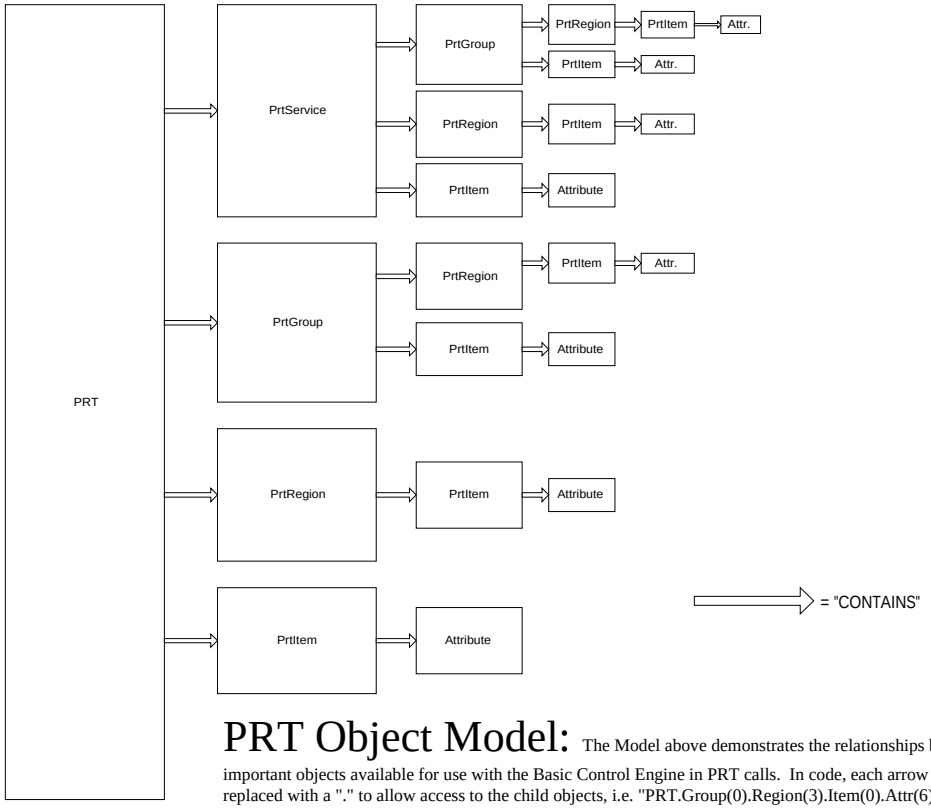
Below is a list of the validations that are performed and the repairs that can be made:

| Problem | Repair Made |
|---|---|
| Cross reference element count larger than slots. | Make number of elements be the number of slots. |
| Cross reference element count larger than number of slots in use. | Make number of elements to be the number of slots in use. |
| More locations are configured than allocated. | None. |
| More locations are occupied than allocated. | None. |
| Fewer items in the region than locations are occupied. | None. |
| More items in the region than possible for the occupied locations. | None. |
| More items in the region than item slots are allocated. | None. |
| The number of items in the region is not consistent with the start and/or end of the location list. | Set the number of items to zero and the location list to empty. |
| Count of an item type is negative. | Make the count zero. |
| Count of all item types does not match the total of items in the region. | None. |
| A region location index is out of range. | None. |
| Location list is circular. | None. |
| An entry of the location list is marked as not being used. | None. |
| An item index is out of range. | None. |
| The count of serialized items does not match the number of items in the serialized item list. | Make the count be the number of items in the serialized item list. |
| The count of non-serialized items does not match the number of items in the non-serialized item list. | Make the count be the number of items in the non-serialized item list. |
| Location list end is at a different spot than is pointed to. | Make the pointer point to the end of the location list. |
| An item is linked into the region twice. | None. |
| An entry on the item list is marked as not being used. | None. |
| An item is on the wrong type of list. | None. |
| An item is in the cross reference but not in the region. | None. |

# Appendix E - Basic Control Engine Extensions for Production Tracking

## PRT Object Model

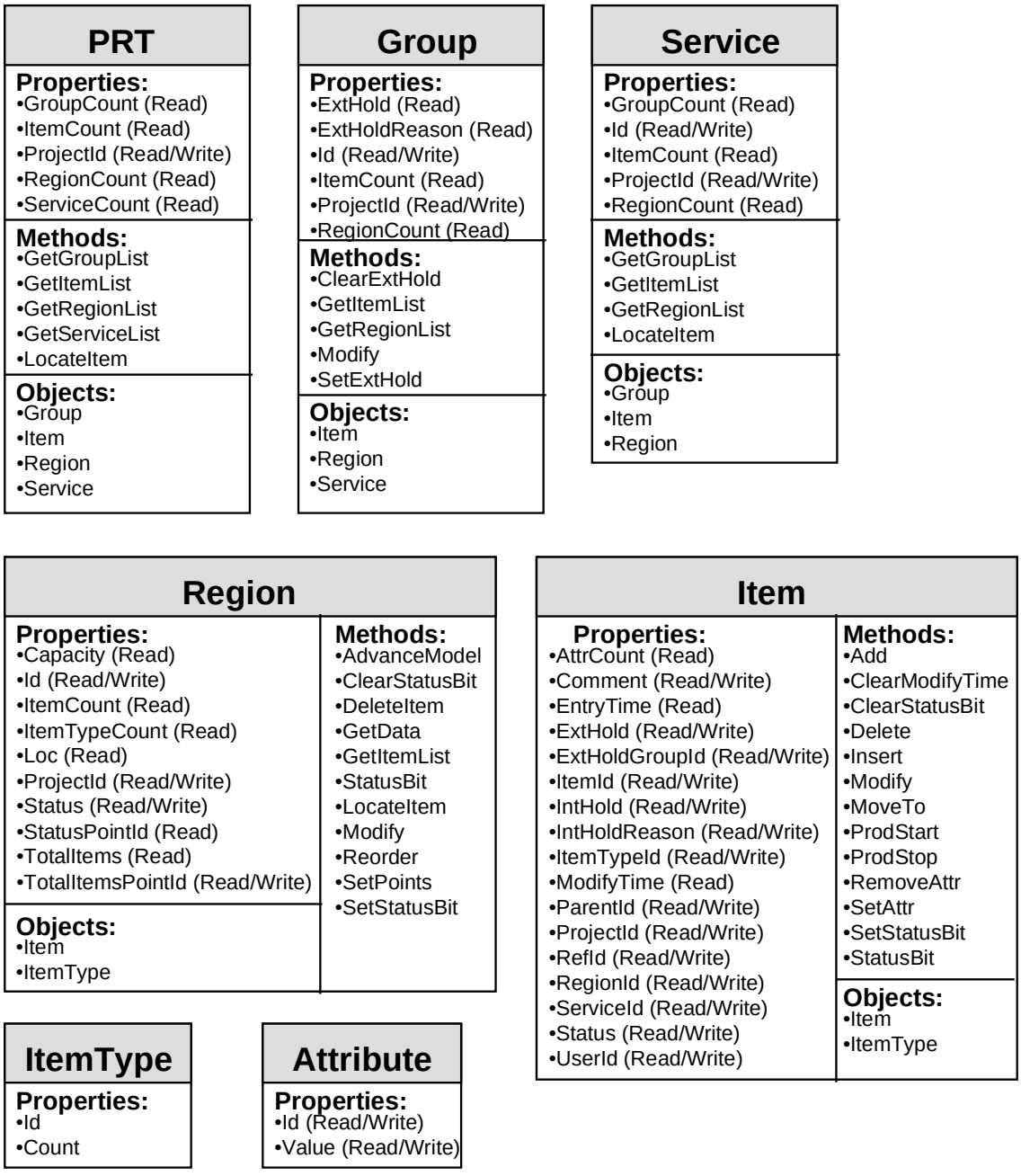
The CIMPLICITY API for Production Tracking (PRT) has been implemented using an object model approach to allow the user to manipulate PRT information in a logical and consistent manner. This section is intended to provide the user with an overview of the PRT object hierarchy and an explanation of the objects' functions.




PRT Object Model: The Model above demonstrates the relationships between the 6 most important objects available for use with the Basic Control Engine in PRT calls. In code, each arrow above would be replaced with a "." to allow access to the child objects, i.e. "PRT.Group(0).Region(3).Item(0).Attr(6).Id".

# PRT Object Attributes

The object attributes for the Basic Control Engine extensions for Production Tracking are:



## Prt

This can be considered the root object of Production Tracking.

From the root you can obtain a list of PRT Services, a list of PRT Groups, a list of PRT Regions, and a list of PRT Items. Obtaining a list of items would return every item currently in the PRT system.

You can also locate an item, which will return all instances of the item within PRT.

## PrtService

This object contains information on a particular PRT Service.

The object can be obtained from a list of PRT Services obtained from the Prt object, or by initializing the object with the appropriate Service ID.

From a Service Object, you can obtain a list of all groups, regions and items contained within the service. You can also locate an item, which will return all instances of the item within PRT Service.

## PrtGroup

This object contains information on the groups configured in your PRT System.

Groups may be places on or off hold via this object. A group object may obtained from a Prt or PrtService List or by initializing the ID (GroupId) property of the object.

From a group object, you can obtain a list of all region and items contained within the group. You can also locate an item, which will return all instances of the item within PRT Group.

## PrtRegion

This object contains a configured PRT region.

Region status, capacity, etc., may be found in this object. A region object may be obtained from a list created by a Prt, PrtService, or PrtGroup object or by setting the ID to a known group ID .

From a region object, you can obtain a list of all items within the region. You can also locate an item, which will return all instances of the item within PRT Region.

## PrtItem

This object contains a configured PRT Item. Item status and attributes may be accessed via this object.

## PrtAttribute

This object contains an attribute for a PRT Item.

# Prt.GetGroupList (Method

**Syntax:**

**`Prt.GetGroupList`**

**Description:**

Gets a list of all PRT groups. After calling the method, the GroupCount and Group routines may be used to access the returned information.)

**Example:**

```
Dim prt as new Prt
' Get a list of groups from PRT
'    and display them one by one in a message box.
prt.GetGroupList
for j = 0 to prt.GroupCount - 1
      MsgBox prt.Group(j).Id
next j
```

# Prt.GetItemList (Method)

**Syntax:**

**`Prt.GetItemList`**

**Description:**

Gets a list of all items in PRT. After calling the method, the total item count can be obtained from `ItemCount` and the individual items may be accessed via the `Item` function.

To get the item list for a particular item class, set Prt.ItemClassId to the desired item class id before calling `Prt.GetItemList`. See Prt.ItemClassId for more details.

**Example:**

```
Dim prt as new Prt
' Get a list of item from PRT
'    and display them one by one in a message box.
prt.GetItemList
for j = 0 to prt.ItemCount - 1
      MsgBox prt.Item(j).ItemId
next j
```

# Prt.GetRegionList (Method)

**Syntax:**

**`Prt.GetRegionList`**

**Description:**

Gets a list of all Tracking regions. After calling the method, the total region count can be obtained from `RegionCount` and the individual regions may be accessed via the `Region` function.

**Example:**

```
Dim prt as new Prt
' Get a list of groups from PRT
'    and display them one by one in a message box.
prt.GetRegionList
for j = 0 to prt.RegionCount - 1
      MsgBox prt.Region(j).Id
next j
```

# Prt.GetServiceList (Method)

**Syntax:**

**`Prt.GetServiceList`**

**Description:**

Gets a list of the configured PRT services. After calling this method, the total number of services can be obtained from `ServiceCount` and the individual services may be accessed via the `Service` function.

**Example:**

```
Dim prt as new Prt
' Get a list of services from PRT
'    and display them one by one in a message box.
prt.GetServiceList
for j = 0 to prt.ServiceCount - 1
      MsgBox prt.Service(j).Id
next j
```

# Prt.Group (Function)

**Syntax:**

**`Prt.Group(index)`**

**Description:**

Returns a group at a specified index from the list was found by a previous call to `GetGroupList`.

**Example:**

```
Dim prt as new Prt
' Get a list of groups from PRT
'    and display them one by one in a message box.
prt.GetGroupList
for j = 0 to prt.GroupCount - 1
      MsgBox prt.Group(j).Id
next j
```

# Prt.GroupCount (Property Read)

**Syntax:**

**`Prt.GroupCount`**

**Description:**

Contains the number of Groups found by a prior call to `GetGroupList`.

**Example:**

```
Dim prt as new Prt
' Get a list of groups from PRT
'    and display them one by one in a message box.
prt.GetGroupList
for j = 0 to prt.GroupCount - 1
      MsgBox prt.Group(j).Id
next j
```

# Prt.Item (Function)

**Syntax:**

**`Prt.Item`**(*index*)

**Description:**

Returns an item at the specified index from the list which was found by a previous call to `GetItemList`.

**Example:**

```
Dim prt as new Prt
' Get a list of item from PRT
'    and display them one by one in a message box.
prt.GetItemList
for j = 0 to prt.ItemCount - 1
      MsgBox prt.Item(j).ItemId
next j
```

# Prt.ItemClassId (Property Read/Write)

**Syntax:**

**`Prt.ItemClassId`**

**Description:**

String property to set or get the Item Class ID in the Prt object. Used in conjunction with the Prt.GetItemList method to retrieve a list of all items of a particular item class in PRT.

**Example:**

```
Dim prt as new Prt
' Get a list of item with a item class id of “CARRIER” from PRT
'    and display them one by one in a message box.
prt.ItemClassId = “CARRIER”
prt.GetItemList
for j = 0 to prt.ItemCount - 1
      MsgBox prt.Item(j).ItemId
next j
```

# Prt.ItemCount (Property Read)

**Syntax:**

**`Prt.ItemCount`**

**Description:**

Contains the number of items find by a previous `GetItemList` or `LocateItem` call.

**Example:**

```
Dim prt as new Prt
' Get a list of item from PRT
'    and display them one by one in a message box.
prt.GetItemList
for j = 0 to prt.ItemCount - 1
      MsgBox prt.Item(j).ItemId
next j
```

# Prt.LocateItem (Method)

**Syntax:**

**`Prt.LocateItem`** *itemId$*, *refId$* [, *numRetries*]

**Description:**

Locates a serialized Item within PRT. On return, the total Item count can be obtained from `ItemCount`, and the individual Items may be accessed via the `Item` function.

**Comments:**

| Parameter | Description |
|---|---|
| *itemId$* | String. The Item ID to locate, may be an empty string if *refId$ is* specified. |
| *refId$* | String. The Reference ID of the item to locate, may omitted or an empty string if *itemId$* is specified. |
| *NumRetries* | Integer (optional) - The number of times to retry to find the item before failing. If an item is in transition from one tracking region to the next, there is a small period of time when it doesn't appear in either regions tracking queue. Defaults to 3 if not specified. |

**Example:**

```
Dim prt as new Prt
' Get a single item from PRT with its RefId
'    and display its ItemId in a message box.
PRT.LocateItem "",REFNUM$
MsgBox prt.Item(0).ItemId
```

# Prt.ProjectId (Property Read/Write)

**Syntax:**

**`Prt.ProjectId`**

**Description:**

String property to set the Project which subsequent Production Tracking extensions will communicate with.

**Example:**

```
Dim prt as new PRT
' Get a list of services from PRT '
    and display them one by one in a message box
prt.ProjectId = "\\project1"
prt.GetServiceList
for j = 0 to prt.ServiceCount - 1
   MsgBox prt.Service(j).Id
next j
```

# Prt.Region (Function)

**Syntax:**

**`Prt.Region(`***index***`)`**

**Description:**

Returns a region at the specified index from the list which was found by a previous call to `GetRegionList`.

**Example:**

```
Dim prt as new Prt
' Get a list of groups from PRT
'    and display them one by one in a message box.
prt.GetRegionList
for j = 0 to prt.RegionCount - 1
      MsgBox prt.Region(j).Id
next j
```

# Prt.RegionCount (Property Read)

**Syntax:**

**`Prt.RegionCount`**

**Description:**

Contains the count of the number of Tracking Regions found by a prior `GetRegionList` call.

**Example:**

```
Dim prt as new Prt
' Get a list of groups from PRT
'    and display them one by one in a message box.
prt.GetRegionList
for j = 0 to prt.RegionCount - 1
MsgBox prt.Region(j).Id
next j
```

# Prt.Service (Function)

**Syntax:**

**`Prt.Service`**

**Description:**

Returns a Service at the specified index from the list which was found by a previous call to `GetServiceList`.

**Example:**

```
Dim prt as new Prt
' Get a list of services from PRT
'    and display them one by one in a message box.
prt.GetServiceList
for j = 0 to prt.ServiceCount - 1
      MsgBox prt.Service(j).Id
next j
```

# Prt.ServiceCount (Property Read)

**Syntax:**

**`Prt.ServiceCount`**

**Description:**

Contains a count of the number of Service found with a prior `GetServiceList` call.

**Example:**

```
Dim prt as new Prt
' Get a list of services from PRT
'    and display them one by one in a message box.
prt.GetServiceList
for j = 0 to prt.ServiceCount - 1
      MsgBox prt.Service(j).Id
next j
```

# PrtAttribute.Id (Property Read/Write)

**Syntax:**

**`PrtAttribute.Id`**

**Description:**

String property contains the attribute name of an Item ID .

**Example:**

```
Dim prt as new Prt
' Get a list of items from PRT and display the first item's
'   attribute names and values one by one in a message box.
prt.GetItemList
for j = 0 to prt.Item(0).AttrCount - 1
  MsgBox prt.Item(0).Attr(j).Id &"="& prt.Item(0).Attr(j).Value
next j
```

# PrtAttribute.Value (Property Read)

**Syntax:**

**PrtAttribute.Value**

**Description:**

String property contains the attribute value of an Item ID.

**Example:**

```
Dim prt as new Prt
' Get a list of items from PRT and display the first item's
'    attribute names and values one by one in a message box.
prt.GetItemList
for j = 0 to prt.Item(0).AttrCount - 1
  MsgBox prt.Item(0).Attr(j).Id &"="& prt.Item(0).Attr(j).Value
next j
```

# PrtCheckRoute (Method)

**Syntax:**

**`PrtCheckRoute`** *sourceRegionId$, destRegionId$, itemtypeId$*

**Description:**

Validates that an Item of a specific type is configured to move from a Source Tracking region to a Destination Tracking Region. Route validation is performed locally by the API from configuration information that was read into the API's internal schema during initialization.

If success the routine returns.

If failure an error is generated and should be caught using the `on error goto` syntax.

**Comments:**

| Parameter | Description |
|---|---|
| *sourceRegionId$* | String. The source region of the item. |
| *destRegionId$* | String. The destination region of the item. |
| *itemtypeId$* | String. The item type ID . |

**Example:**

```
RtFail = 0
On Error goto RouteFail
PrtCheckRoute "Schedule", "Production", "MARKV"
If RtFail = 0 then msgbox "Item Type Valid for this Route."
If RtFail = 1 then msgbox "Item Type Invalid for this Route!"
Exit Sub
RouteFail:
    RtFail = 1
    Resume Next
```

# PrtGroup.ClearExtHold (Method

**Syntax:**

**`PrtGroup.ClearExtHold`**

**Description:**

Clears an external hold for this tracking group. This setting will be updated to PRT after a call to `Modify`.)

**Example:**

```
Dim prt as new Prt
' Get a list of groups from PRT
'    and clear each one's External Hold status bit.
prt.GetGroupList
for j = 0 to prt.GroupCount - 1
       prt.Group(j).ClearExtHold
       prt.Group(j).Modify
next j
```

# PrtGroup.ExtHold (Property Read)

**Syntax:**

**`PrtGroup.ExtHold`**

**Description:**

Boolean property contains the setting of External Hold for the Tracking Group.

**Example:**

```
Dim prt as new Prt
' Get a list of groups from PRT and display their External Hold
'    status bits one by one in a message box.
prt.GetGroupList
for j = 0 to prt.GroupCount - 1
      MsgBox prt.Group(j).ExtHold
next j
```

# PrtGroup.ExtHoldReason (Property Read)

**Syntax:**

**`PrtGroup.ExtHoldReason`**

**Description:**

String property contains the reason for setting External Hold for this Tracking Group.

**Example:**

```
Dim prt as new Prt
' Get a list of groups from PRT and display their External Hold
'    status bit Reasons one by one in a message box.
prt.GetGroupList
for j = 0 to prt.GroupCount - 1
      MsgBox prt.Group(j).ExtHoldReason
next j
```

# PrtGroup.GetItemList (Method)

**Syntax:**

**`PrtGroup.GetItemList`**

**Description:**

Gets a list of all items in the Tracking Group. On return, the total item count can be obtained from `PrtGroup.ItemCount` and the individual items may be accessed via the `PrtGroup.Item` function.

To get the item list for a particular item class, set PrtGroup.ItemClassId to the desired item class id before calling `PrtGroup.GetItemList`. *See PrtGroup.ItemClassId for more details.*

**Example:**

```
Dim main as new Prtgroup
main.Id = "MAIN"
' Get a list of items from group "MAIN"
'    and display them one by one in a message box.
main.GetItemList
for j = 0 to main.ItemCount - 1
      MsgBox main.Item(j).ItemId
next j
```

# PrtGroup.GetRegionList (Method)

**Syntax:**

**`PrtGroup.GetRegionList`**

**Description:**

Gets a list of Tracking regions contained in the Tracking Group. On return, the total number of regions can be obtained from `PrtGroup.RegionCount` and the individual items may be accessed via the `PrtGroup.Region` function.

**Example:**

```
Dim main as new PrtGroup
main.Id = "MAIN"
' Get a list of regions from a PRT group
'    and display them one by one in a message box.
main.GetRegionList
for j = 0 to main.RegionCount - 1
      MsgBox main.Region(j).Id
next j
```

# PrtGroup.Id (Property Read/Write)

**Syntax:**

**`PrtGroup.Id`**

**Description:**

String property contains the Group Id of the object.

**Example:**

```
Dim main as new PrtGroup
main.Id = "MAIN"
' Get a list of regions from a PRT group
'    and display them one by one in a message box.
main.GetRegionList
for j = 0 to main.RegionCount - 1
      MsgBox main.Region(j).Id
next j
```

# PrtGroup.Item (Function)

**Syntax:**

**`PrtGroup.Item(`***index***`)`**

**Description:**

Returns an item at the specified index from the list which was found by a previous call to `GetItemList`.

**Example:**

```
Dim main as new Prtgroup
main.Id = "MAIN"
' Get a list of items from group "MAIN"
'    and display them one by one in a message box.
main.GetItemList
for j = 0 to main.ItemCount - 1
      MsgBox main.Item(j).ItemId
next j
```

# PrtGroup.ItemClassId (Property Read/Write)

**Syntax:**

**PrtGroup.ItemClassId**

**Description:**

String property to set or get the Item Class ID in the PrtGroup object. Used in conjunction with the PrtGroup.GetItemList method to retrieve a list of all items of a particular item class in a Tracking Group.

**Example:**

```
Dim main as new Prtgroup
main.Id = "MAIN"

' Get a list of items with an item class id “CARRIER” from the
'    group "MAIN" and display them one by one in a message box.
main.ItemClassId = “CARRIER”
main.GetItemList
for j = 0 to main.ItemCount - 1
      MsgBox main.Item(j).ItemId
next j
```

# PrtGroup.ItemCount (Property Read)

**Syntax:**

**PrtGroup.ItemCount**

**Description:**

Integer property contains the number of items find by a previous `GetItemList` or `LocateItem` call.

**Example:**

```
Dim main as new Prtgroup
main.Id = "MAIN"
' Get a list of items from group "MAIN"
'    and display them one by one in a message box.
main.GetItemList
for j = 0 to main.ItemCount - 1
      MsgBox main.Item(j).ItemId
next j
```

# PrtGroup.LocateItem (Method)

**Syntax:**

**`PrtGroup.LocateItem`** *itemId$, refId$* **[**, *numRetries***]**

**Description:**

Locates a serialized item within the tracking group. Upon return, the total item count may be obtained from `ItemCount`, and the individual items may be accessed via the `Item` function.

**Comments:**

| Parameter | Description |
|---|---|
| *itemId$* | String. The Item ID to locate, may be an empty string if *refId$* is specified. |
| *refId$* | String. The Reference ID of the item to locate, may be omitted or an empty string if *itemId$* is specified. |
| *numRetries* | Integer (optional) - The number of times to retry to find the item before failing. If an item is in transition from one tracking region to the next, there is a small period of time when it doesn't appear in either region's tracking queue. Defaults to 3 if not specified. |

**Example:**

```
Dim main as new PrtGroup
main.Id = "MAIN"
' Get a single item from PRT with its RefId
'    and display its ItemId in a message box.
PRT.LocateItem "",REFNUM$
MsgBox prt.Item(0).ItemId
```

# PrtGroup.Modify (Method)

**Syntax:**

**`PrtGroup.Modify`**

**Description:**

Updates the PRT Tracking Group with the current contents of this object. Used to set or clear External Hold.

**Example:**

```
Dim prt as new Prt
' Get a list of groups from PRT
'    and clear each one's External Hold status bit.
prt.GetGroupList
for j = 0 to prt.GroupCount - 1
       prt.Group(j).ClearExtHold
       prt.Group(j).Modify
next j
```

# PrtGroup.ProjectId (Property Read/Write)

**Syntax:**

**`PrtGroup.ProjectId`**

**Description:**

String property to set the project that subsequent Production Tracking extensions will communicate with.

**Example:**

```
Dim prt as new PrtGroup
' Get a list of services from PRT
'    and display them one by one in a message box.
prt.ProjectId = "\\project1"
prt.GetRegionList
for j = 0 to prt.RegionCount - 1
       MsgBox prt.Region(j).Id
next j
```

# PrtGroup.Region (Function)

**Syntax:**

**`PrtGroup.Region(`***index***`)`**

**Description:**

Returns a region at the specified index from the list that was found by a previous call to `GetRegionList`.

**Example:**

```
Dim main as new PrtGroup
main.Id = "MAIN"
' Get a list of regions from a PRT group
'    and display them one by one in a message box.
main.GetRegionList
for j = 0 to main.RegionCount - 1
      MsgBox main.Region(j).Id
next j
```

# PrtGroup.RegionCount (Property Read)

**Syntax:**

**`PrtGroup.RegionCount`**

**Description:**

Integer property contains a count of the number of Tracking Regions found by a prior `GetRegionList` call.

**Example:**

```
Dim main as new PrtGroup
main.Id = "MAIN"
' Get a list of regions from a PRT group
'    and display them one by one in a message box.
main.GetRegionList
for j = 0 to main.RegionCount - 1
      MsgBox main.Region(j).Id
next j
```

# PrtGroup.SetExtHold (Method)

**Syntax:**

**`PrtGroup.SetExtHold`** *reason$*

**Description:**

Sets an External Hold for this tracking group. The setting will be updated to PRT after a call to `Modify`.

**Comments:**

| Parameter | Description |
|---|---|
| *reason$* | String. The reasons for placing the group on hold. |

**Example:**

```
Dim prt as new Prt
Reason$ = "-----"
' Get a list of groups from PRT
'    and initialize their External Hold status bits and Reasons.
prt.GetGroupList
for j = 0 to prt.GroupCount - 1
       prt.Group(j).SetExtHold Reason$ & " " & j
       prt.Group(j).Modify
next j
```

# PrtItem.Add (Method)

**Syntax:**

**`PrtItem.Add [`***extProcFlag***`]`**

**Description:**

Adds an individual item at the same location where another item currently resides. The `PrtItem.RegionLoc` indicates the location at which the item will be added. The `RegionLoc` can be specified as `PRT_FIRST, PRT_LAST`, or a valid region location within the region.

**Comments:**

| Parameter | Description |
|---|---|
| *ext_proc_flag* | Boolean. External process flag, if set to TRUE; get all information from an external process. If not provided defaults to FALSE. |

If the new item is not being copied from a previously created item, the following **`PRTItem`** object methods must be set for the add to be successful:

**`.RefId`** or **`.ItemId`** (if this is a serialized item)

**`.ItemType`**

**`.RegionId`**

**`.Status`**

**`.IntHold`** (usually false)

**`.ExtHold`** (usually false)

**`.RegionLoc`**

**Example:**

```
dim NEWITEM as new prtitem
'Initialize the new item structure
NEWITEM.ItemId = IDNUM$
NEWITEM.GroupId = Groupname$
NEWITEM.RegionId = Regname$
NEWITEM.RegionLoc = PRT_LAST
NEWITEM.IntHold false
NEWITEM.ExtHold false
NEWITEM.ItemTypeId = TYPE$
NEWITEM.Status = 0
'Upload to Tracking Database
NEWITEM.Add
```

# PrtItem.Attr (Function)

**Syntax:**

**PrtItem.Attr(***index***)**

**PrtItem.Attr(***attrib$***)**

**Description:**

Prototype 1.

Returns an Item Attribute at the specified index from the list which was found by a previous call to `GetItemList`. The index must be in range from `0` to `AttrCount`. If the index is out of range, an error is generated.

Prototype 2.

This function returns the Item Attribute associated with the specified PRT Attribute or an empty string if the Item does not have the Attribute.

**Example:**

```
Dim prt as new Prt
'Get a list of items from PRT and display each of their
'attributes one by one in a message box.
prt.GetItemList
For i = 0 To prt.ItemCount - 1
    For j = 0 To prt.Item(i).AttrCount - 1
        MsgBox prt.Item(i).Attr(j).Id & " = " &
        prt.Item(i).Attr(j).Value
    Next j
Next i
'Get a list of items from PRT again but display their COLOR only
For i = 0 To prt.ItemCount - 1
    For j = 0 To prt.Item(i).AttrCount - 1
        MsgBox prt.Item(i).Attr("COLOR")
    Next j
Next i
```

# PrtItem.AttrCount (Property Read)

**Syntax:**

**PrtItem.AttrCount**

**Description:**

Integer property contains a count of the number of Attributes for the Item.

**Example:**

```
Dim prt as new Prt
'Get a list of items from PRT and display each of their
'attributes one by one in a message box.
prt.GetItemList
For i = 0 To prt.ItemCount - 1
    For j = 0 To prt.Item(i).AttrCount - 1
        MsgBox prt.Item(i).Attr(j).Id & " = " &
      prt.Item(i).Attr(j).Value
    Next j
Next i
'Get a list of items from PRT again but display their COLOR only
For i = 0 To prt.ItemCount - 1
    For j = 0 To prt.Item(i).AttrCount - 1
        MsgBox prt.Item(i).Attr("COLOR")
    Next j
Next i
```

# PrtItem.ClearModifyTime (Method)

**Syntax:**

**PrtItem.ClearModifyTime**

**Description:**

Clears the modify time on the object. When calling `Modify`, PRT will reject the request if the item's last modify time is less than the current modify time of the item in PRT. Calling `ClearModifyTime` overrides this behavior. This should be used with care, since you may overwrite properties of the object that the user may have manipulated through PRT_UI, etc.

**Example:**

```
Dim prt as new Prt
' Get a list of items from PRT
'    and initialize the modify time for each.
prt.GetItemList
for j = 0 to prt.ItemCount - 1
        prt.Item(j).ClearModifyTime
        prt.Item(j).Modify
next j
```

# PrtItem.ClearStatusBit (Property Read/Write

**Syntax:**

```
PrtItem.ClearStatusBit Bit%
```

**Description:**

Clears the specified bit (0 through 31) of the status mask.)

**Comments:**

| Parameter | Description |
|---|---|
| *Bit* | Integer. The bit position 0 - 31 to clear. |

**Example:**

```
Dim prt as new Prt
n% = 17
' Get a list of items from PRT and initialize a particular
'    status bit for each one.
prt.GetItemList
for j = 0 to prt.ItemCount - 1
       prt.Item(j).ClearStatusBit n%
       prt.Item(j).Modify
next j
```

# PrtItem.Comment (Property Read/Write)

**Syntax:**

**`PrtItem.Comment`**

**Description:**

String property to set or get the comment associated with the item for an impending `Modify` or `Add`. The comment is written to the PRT log file as the modifications are sent to the object.

**Example:**

```
dim NEWITEM as new prtitem
' Initialize the new item structure
NEWITEM.ItemId = IDNUM$
NEWITEM.GroupId = Groupname$
NEWITEM.RegionId = Regname$
NEWITEM.RegionLoc = PRT_LAST
NEWITEM.IntHold false
NEWITEM.ExtHold false
NEWITEM.ItemTypeId = TYPE$
NEWITEM.Status = 0
NEWITEM.Comment = "New item created."
' Upload to Tracking Database
NEWITEM.Add
```

# PrtItem.Delete (Property Read/Write)

**Syntax:**

**PrtItem.Delete** *disposition* **[**, *comment$***]**

**Description:**

Deletes the item from the tracking model. A disposition must be supplied, and a comment is optional.

**Comments:**

| Parameter | Description |
|---|---|
| *disposition* | Integer. Reason for deleting. See table below for manifest constants. |
| *comment$* | String (optional) - The optional comment to send to the log file. |

| Disposition | Description |
|---|---|
| PRT_SCRAP | Scrap the item. |
| PRT_DELETE | Delete the item. |
| PRT_NO_CHECK_SCRAP | Scrap the item overriding region locking status |
| PRT_NO_CHECK_DELETE | Delete the item overriding region locking status |

**Example:**

```
Dim prt as new Prt
' Get a single item from PRT with its RefId and delete it.
PRT.LocateItem "",REFNUM$
prt.Item(0).Delete PRT_NO_CHECK_DELETE, "Item Removed"
```

# PrtItem.EntryTime (Property Read)

**Syntax:**

**PrtItem.EntryTime**

**Description:**

Date property contains the date and time at which the item entered the current region.

**Example:**

```
Dim Region as new prt.Region
Region.Id = "Detainment"
'Get a list of items in a region and display their entry times
'    one by one in a message box.
Region.GetItemList
For j = 0 to Region.ItemCount - 1
       msgbox Region.Item(j).EntryTime
Next j
```

# PrtItem.ExtHold (Property Read/Write)

**Syntax:**

**PRTItem.ExtHold [***True/False***]**

**Description:**

Boolean property to set or get the state of External Hold for the Item.

**Example:**

```
dim NEWITEM as new prtitem
'Initialize the new item structure
NEWITEM.ItemId = IDNUM$
NEWITEM.GroupId = Groupname$
NEWITEM.RegionId = Regname$
NEWITEM.RegionLoc = PRT_LAST
NEWITEM.IntHold false
NEWITEM.ExtHold false
NEWITEM.ItemTypeId = TYPE$
NEWITEM.Status = 0
'Upload to Tracking Database
NEWITEM.Add
```

# PrtItem.ExtHoldGroupId(Property Read/Write)

**Syntax:**

**PrtItem.ExtHoldGroupId**

**Description:**

String property to set or get the Group ID which has placed an External Hold on the item.

**Example:**

```
dim NEWITEM as new prtitem
'Initialize the new item structure
NEWITEM.ItemId = IDNUM$
NEWITEM.GroupId = Groupname$
NEWITEM.RegionId = Regname$
NEWITEM.RegionLoc = PRT_LAST
NEWITEM.IntHold false
NEWITEM.ExtHold true
NEWITEM.ExtHoldGroupId = "SHIPPING"
NEWITEM.ItemTypeId = TYPE$
NEWITEM.Status = 0
'Upload to Tracking Database
NEWITEM.Add
```

# PrtItem.ItemClassId (Property Read/Write)

**Syntax:**

**PrtItem.ItemClassId**

**Description:**

String property to set or get the Item Class ID of the Item.

**Example:**

```
dim NEWITEM as new prtitem
'Initialize the new item structure
NEWITEM.ItemId = IDNUM$
NEWITEM.GroupId = Groupname$
NEWITEM.RegionId = Regname$
NEWITEM.RegionLoc = PRT_LAST
NEWITEM.IntHold false
NEWITEM.ExtHold false
NEWITEM.ItemClassId = CLASS$
NEWITEM.Status = 0
'Upload to Tracking Database
NEWITEM.Add
```

# PrtItem.ItemId (Property Read/Write)

**Syntax:**

**`PrtItem.ItemId`**

**Description:**

String property to set or get the Item ID of the Item in the region. Setting has does not update PRT until the `Modify` method is invoked.

**Example:**

```
dim NEWITEM as new prtitem
'Initialize the new item structure
NEWITEM.ItemId = IDNUM$
NEWITEM.GroupId = Groupname$
NEWITEM.RegionId = Regname$
NEWITEM.RegionLoc = PRT_LAST
NEWITEM.IntHold false
NEWITEM.ExtHold false
NEWITEM.ItemTypeId = TYPE$
NEWITEM.Status = 0
'Upload to Tracking Database
NEWITEM.Add
```

# PrtItem.Insert (Method)

**Syntax:**

**PrtItem.Insert [***extProcFlag***]**

**Description:**

Inserts an Item into the Tracking Region queue. If an Item is to be inserted into the queue at a particular location (other than at the head or tail of the queue), the Region location of the Item to be inserted ahead of is specified, i.e. specifying `regionLoc = 1`, is equivalent to specifying `regionLoc = PRT_FIRST`.

**Comments:**

| Parameter | Description |
|---|---|
| *ext_proc_flag* | Boolean. External process flag, if set to TRUE; get all information from an external process. If not provided defaults to FALSE. |

**Example:**

```
dim NEWITEM as new prtitem
'Initialize the new item structure
NEWITEM.ItemId = IDNUM$
NEWITEM.GroupId = Groupname$
NEWITEM.RegionId = Regname$
NEWITEM.RegionLoc = 3
NEWITEM.IntHold false
NEWITEM.ExtHold false
NEWITEM.ItemTypeId = TYPE$
NEWITEM.Status = 0
'Upload to Tracking Database
NEWITEM.Insert
```

# PrtItem.IntHold (Property Read/Write)

**Syntax:**

**PrtItem.IntHold**

**Description:**

Boolean property to set or get the state of Internal Hold on the Item.

**Example:**

```
dim NEWITEM as new prtitem
'Initialize the new item structure
NEWITEM.ItemId = IDNUM$
NEWITEM.GroupId = Groupname$
NEWITEM.RegionId = Regname$
NEWITEM.RegionLoc = PRT_LAST
NEWITEM.IntHold false
NEWITEM.ExtHold false
NEWITEM.ItemTypeId = TYPE$
NEWITEM.Status = 0
'Upload to Tracking Database
NEWITEM.Add
```

# PrtItem.IntHoldReason (Property Read/Write)

**Syntax:**

**`PrtItem.IntHoldReason`**

**Description:**

String property to set or get the Internal Hold Reason for the Item.

**Example:**

```
dim NEWITEM as new prtitem
'Initialize the new item structure
NEWITEM.ItemId = IDNUM$
NEWITEM.GroupId = Groupname$
NEWITEM.RegionId = Regname$
NEWITEM.RegionLoc = PRT_LAST
NEWITEM.IntHold true
NEWITEM.IntHoldReason = "Insufficient Parts"
NEWITEM.ExtHold false
NEWITEM.ItemTypeId = TYPE$
NEWITEM.Status = 0
'Upload to Tracking Database
NEWITEM.Add
```

# PrtItem.ItemTypeId (Property Read/Write)

**Syntax:**

**PrtItem.ItemTypeId**

**Description:**

String property to set or get the Item Type ID of the Item.

**Example:**

```
dim NEWITEM as new prtitem
'Initialize the new item structure
NEWITEM.ItemId = IDNUM$
NEWITEM.GroupId = Groupname$
NEWITEM.RegionId = Regname$
NEWITEM.RegionLoc = PRT_LAST
NEWITEM.IntHold false
NEWITEM.ExtHold false
NEWITEM.ItemTypeId = TYPE$
NEWITEM.Status = 0
'Upload to Tracking Database
NEWITEM.Add
```

# PrtItem.Modify (Method)

**Syntax:**

**PrtItem.Modify**

**Description:**

Modifies the Item. All modified attributes and properties of the item are sent to the PRT Data Server. An error will be generated if the PRT Data Server rejects the request.

**Example:**

```
Dim prt as new Prt
n% = 17
' Get a list of items from PRT and initialize a particular
'    status bit for each one.
prt.GetItemList
for j = 0 to prt.ItemCount - 1
       prt.Item(j).ClearStatusBit n%
       prt.Item(j).Modify
next j
```

# PrtItem.ModifyTime (Property Read)

**Syntax:**

**`PrtItem.ModifyTime`**

**Description:**

Date property contains the time at which the Item was modified. When the user modifies the Item, PRT checks this modify time against the last modify time of the Item. If the modify time is older than the current modify time of the Item in PRT, PRT will not allow the user to change the Item. This is to prevent two processes from modifying an Item without each being aware of the other's modification. To override this behavior, call `ClearModifyTime` which will disable this checking.

**Example:**

```
Dim Region as new prt.Region
Region.Id = "Detainment"
'Get a list of items in a region and display their modify times
'    one by one in a message box.
Region.GetItemList
For j = 0 to Region.ItemCount - 1
       msgbox Region.Item(j).ModifyTime
Next j
```

# PrtItem.MoveTo (Method)

**Syntax:**

**`PrtItem.MoveTo`** *destRegionId$, destLoc, insertFlag* **[**, *comment$***]**

**Description:**

Moves an item from its current location to a new location in a region (could be the same region.)

**Comments:**

| Parameter | Description |
|---|---|
| *destRegionId$* | String. Destination Region ID. The source Region ID is contained in the item object already. |
| *destLoc* | Integer. Location to move to in destination region. Sequence number of the item in the destination tracking region. Specified as PRT_FIRST (EXIT) or PRT_LAST (Entry), or a number from 1 to n, where n is the number of items in the queue. |
| *insert_flag* | Boolean. If true, the moved item is to be inserted in the destination region queue before the item located at the specified destination region location, unless the manifest constant last is specified, in which case the item is inserted after (behind) the last item currently present in the region. |
| *comment$* | String (optional). An optional comment to write to the PRT log file. |

**Example:**

```
Dim Region as new PRTREGION
Region.Id = SRC$
Region.GetItemList
' Set timestamp attributes for the head item in a region, and
'    then move the item to another region.
DSUF$ = "_DATE"
TSUF$ = "_TIME"
Region.Item(0).Setattr SRC$ & DSUF$, DATE$
Region.Item(0).Setattr SRC$ & TSUF$, TIME$
Region.Item(0).Modify
Region.Item(0).Moveto DESTREGION$, RegionLOC%, 0
```

# PrtItem.ParentId (Property Read/Write)

**Syntax:**

**`PrtItem.ParentId`**

**Description:**

String property to set or get the Parent ID of the item. Parent IDs are only valid for non-serialized items.

**Example:**

```
Dim Region as new PRTRegion
Region.Id = "DETAINMENT"
' Get a list of all the items which exist in 1 location of a
'    region and display the ParentIds one by one in a message box.
Region.Loc = 0
Region.GetItemList
for j = 0 to Region.ItemCount - 1
       msgbox Region.Item(j).ParentId
next j
```

# PrtItem.ProdStart (Method)

**Syntax:**

**PrtItem.ProdStart [***extProcFlag***]**

**Description:**

Initiates production start for an item in a Tracking Region. It is equivalent to the Add method, but logs differently.

**Comments:**

| Parameter | Description |
|---|---|
| *ext_proc_flag* | Boolean. External process flag, if set to TRUE, get all information from an external process. If not provided defaults to FALSE. |

**Example:**

```
dim NEWITEM as new prtitem
'Initialize the new item structure
NEWITEM.ItemId = IDNUM$
NEWITEM.GroupId = Groupname$
NEWITEM.RegionId = Regname$
NEWITEM.RegionLoc = PRT_LAST
NEWITEM.IntHold false
NEWITEM.ExtHold false
NEWITEM.ItemTypeId = TYPE$
NEWITEM.Status = 0
'Upload to Tracking Database
NEWITEM.ProdStart
```

# PrtItem.ProdStop (Method)

**Syntax:**

**`PrtItem.ProdStop`**

**Description:**

Removes an Item from the Production Tracking system as the result of its exiting out of a region along a route that has no destination region. The specified Item must currently reside at a source region associated with such a route. If the Item cannot legitimately transition out of the system along such a route, an error is returned to the calling routine.

**Example:**

```
Dim Region as new prt.Region
Region.Id = "Detainment"
'Get a list of items in a region and remove each of them from
'    tracking one by one.
Region.GetItemList
For j = 0 to Region.ItemCount - 1
       Region.Item(j).ProdStop
Next j
```

# PrtItem.ProjectId (Property Read/Write)

**Syntax:**

**PrtItem.ProjectId**

**Description:**

String property to set the project with which subsequent Production Tracking extensions will communicate.

**Example:**

```
Dim prt as new PrtItem
' Get a list of services from PRT and display them one by one
'    in a message box
prt.ProjectId = "\\project1"
for j = 0 to 19
        index$ = CSTR(j)
        prt.SetAttr "NAME" & index$, "Val" & index$
next j
```

# PrtItem.RefId (Property Read/Write)

**Syntax:**

**PrtItem.RefId**

**Description:**

String property to set or get the Reference ID of the Item.

**Example:**

```
dim NEWITEM as new prtitem
'Initialize the new item structure
NEWITEM.ItemId = IDNUM$
NEWITEM.RefId = RefNUM$
NEWITEM.GroupId = Groupname$
NEWITEM.RegionId = Regname$
NEWITEM.RegionLoc = PRT_LAST
NEWITEM.IntHold false
NEWITEM.ExtHold false
NEWITEM.ItemTypeId = TYPE$
NEWITEM.Status = 0
'Upload to Tracking Database
NEWITEM.ProdStart
```

# PrtItem.RegionId (Property Read/Write)

**Syntax:**

**`PrtItem.RegionId`**

**Description:**

String property to set or get the Region ID of the Item. Setting the Region ID is not updated into PRT until the Item is modified.

**Example:**

```
dim NEWITEM as new prtitem
'Initialize the new item structure
NEWITEM.ItemId = IDNUM$
NEWITEM.RefId = RefNUM$
NEWITEM.GroupId = Groupname$
NEWITEM.RegionId = Regname$
NEWITEM.RegionLoc = PRT_LAST
NEWITEM.IntHold false
NEWITEM.ExtHold false
NEWITEM.ItemTypeId = TYPE$
NEWITEM.Status = 0
'Upload to Tracking Database
NEWITEM.Add
```

# PrtItem.RegionLoc (Property Read/Write)

**Syntax:**

**`Point.RegionLoc`**

**Description:**

Integer property to set or get the location of the Item in the region. Setting does not update PRT until the `Modify` method is invoked.

**Example:**

```
dim NEWITEM as new prtitem
'Initialize the new item structure
NEWITEM.ItemId = IDNUM$
NEWITEM.RefId = RefNUM$
NEWITEM.GroupId = Groupname$
NEWITEM.RegionId = Regname$
NEWITEM.RegionLoc = PRT_LAST
NEWITEM.IntHold false
NEWITEM.ExtHold false
NEWITEM.ItemTypeId = TYPE$
NEWITEM.Status = 0
'Upload to Tracking Database
NEWITEM.Add
```

# PrtItem.RemoveAttr(Method)

**Syntax:**

**`PrtItem.RemoveAttr`** *attrib$*

**Description:**

Removes an Attribute from the Item. If the Attribute is not found an error is generated.

**Example:**

```
Dim Region as new prt.Region
Region.Id = "Detainment"
'Get a list of items in a region and remove an attribute from
'    each of them one by one.
Region.GetItemList
For j = 0 to Region.ItemCount - 1
       Region.Item(j).RemoveAttr "BODYCOLOR"
Next j
```

# PrtItem.ServiceId (Property Read/Write)

**Syntax:**

**`PrtItem.ServiceId`**

**Description:**

String property to get or set the Service ID associated with the Item.

**Example:**

```
dim NEWITEM as new prtitem
'Initialize the new item structure
NEWITEM.ItemId = IDNUM$
NEWITEM.ServiceId = "PRT_DC"
NEWITEM.GroupId = Groupname$
NEWITEM.RegionId = Regname$
NEWITEM.RegionLoc = PRT_LAST
NEWITEM.IntHold false
NEWITEM.ExtHold false
NEWITEM.ItemTypeId = TYPE$
NEWITEM.Status = 0
'Upload to Tracking Database
NEWITEM.Add
```

# PrtItem.SetAttr (Method)

**Syntax:**

**PrtItem.SetAttr** *attrib$, value$*

**Description:**

Sets an Attribute to a value for the Item. If the Attribute exists for the Item, it is overwritten. If the Attribute does not exist for the Item a new Attribute is created. PRT will be updated with the modified Attribute on the next `Modify` method.

**Example:**

```
Dim Region as new PRTREGION
Region.Id = SRC$
Region.GetItemList
'Set timestamp attributes for the head item in a region, and
'   then move the item to another region.
DSUF$ = "_DATE"
TSUF$ = "_TIME"
Region.Item(0).Setattr SRC$ & DSUF$, DATE$
Region.Item(0).Setattr SRC$ & TSUF$, TIME$
Region.Item(0).Modify
Region.Item(0).Moveto DESTREGION$, RegionLOC%, 0
```

# PrtItem.SetStatusBit (Method)

**Syntax:**

**PrtItem.SetStatusBit** *Bit%*

**Description:**

Sets a specific bit (0 through 31) of the Status Mask for an Item.

**Example:**

```
Dim CurPrt as new prtregion
CurPrt.Id = "EXITREGION"
CurPrt.GetItemList
'Get a list of items in a region, check each one's status bit,
'    then set it if it's not already set.
If CurPrt.Item(0).StatusBit(J%) = FALSE Then
        CurPrt.Item(I).SetStatusBit J%
        CurPrt.Item(I).Modify
End If
```

# PrtItem.Status (Property Read/Write)

**Syntax:**

**PrtItem.Status**

**Description:**

Long property to get or set the Status Mask of the Item.

**Example:**

```
dim NEWITEM as new prtitem
'Initialize the new item structure
NEWITEM.ItemId = IDNUM$
NEWITEM.GroupId = Groupname$
NEWITEM.RegionId = Regname$
NEWITEM.RegionLoc = PRT_LAST
NEWITEM.IntHold false
NEWITEM.ExtHold false
NEWITEM.ItemTypeId = TYPE$
NEWITEM.Status = 0
'Upload to Tracking Database
NEWITEM.Add
```

# PrtItem.StatusBit (Function)

**Syntax:**

**PrtItem.StatusBit** *bit%*

**Description:**

Retrieves the setting of a specific status bit.

**Comments:**

| Parameter | Description |
|---|---|
| *bit* | Integer. 0-31 bit position to return. |

**Example:**

```
Dim CurPrt as new prtregion
CurPrt.Id = "EXITREGION"
CurPrt.GetItemList
'Get a list of items in a region, check each one's status bit,
'    then set it if it's not already set.
If CurPrt.Item(0).StatusBit(J%) = FALSE Then
       CurPrt.Item(I).SetStatusBit J%
       CurPrt.Item(I).Modify
End If
```

# PrtItem.UserId (Property Read/Write)

**Syntax:**

**PrtItem.UserId**

**Description:**

String property to set or get the User ID associated with the Item. The User ID is recorded into the PRT log when an Item is modified.

**Example:**

```
Dim prt as new Prt
n% = 17
' Get a list of items from PRT and initialize a particular
'    status bit for each one, setting the object to log the
'    user id performing the action.
prt.GetItemList
for j = 0 to prt.ItemCount - 1
       prt.Item(j).ClearStatusBit n%
       prt.Item(j).UserId = "ADMINISTRATOR"
       prt.Item(j).Modify
next j
```

# PrtRegion.AdvanceModel (Method)

**Syntax:**

**`PrtRegion.AdvanceModel`** *destRegionId$* **[**, *comment$* **[**, *userId$***]**

**Description:**

Advances the next Item Carrier to leave the region to a destination region.

**Note:** This call moves all items at the FIRST (EXIT) location in the source region to the LAST (ENTRY) location in the destination region.

**Comments:**

| Parameter | Description |
|---|---|
| *destRegionId$* | String. Destination Region ID; region the item is moving into. |
| *comment$* | String. (Optional) Comment to place in PRT log file. |
| *userId$* | String. (Optional) User ID to place into PRT log file. |

**Example:**

```
Dim Region as new PrtRegion
Region.Id = "Schedule"
'Determine if there are any items in the region,
'    and advance the first one if there are.
Region.GetItemList
If Region.ItemCount > 0 then
       Region.AdvanceModel "Production"
End If
```

# PrtRegion.Capacity (Property Read)

**Syntax:**

**`PrtRegion.Capacity`**

**Description:**

Integer property to return the region capacity.

**Example:**

```
Dim Prt as new Prt
Prt.GetRegionList
For i = 0 to Prt.RegionCount - 1
       msgbox Prt.Region(i).Capacity
Next i
```

# PrtRegion.ClearStatusBit (Method)

**Syntax:**

**`PrtRegion.ClearStatusBit`** *b%*

**Description:**

Clears a particular status bit in the regions status mask.

**Comments:**

To obtain the region's status mask, use the `PrtRegion.GetData` method. To update the Region's status mask to PRT, use the `PrtRegion.Modify`.

Bits 0-15 are for User Use. Bits 16-31 are for GE Fanuc use. The following constants may be used to access GE Fanuc specific bits.

| Constants | Description |
|---|---|
| PRT_REGION_FULL | Region Full Bit |
| PRT_REGION_EXT_HOLD | Region External Hold Bit |
| PRT_REGION_IN_LOCK | Region in Lock Bit |
| PRT_REGION_OUT_LOCK | Region Out Lock Bit |
| PRT_REGION_OUT_OF_SEQ | Region Out of Sequence Bit |
| PRT_REGION_CAP_EXCEEDED | Region Capacity Exceeded Bit |
| PRT_REGION_NORMAL | Region Normal Bit |

**Example:**

```
Dim Prt as new Prt
n% = 17
' Get a list of items from PRT and initialize a particular
'    status bit for each one.
Prt.GetRegionList
for j = 0 to prt.ItemCount - 1
       Prt.Region(j).ClearStatusBit n%
       Prt.Region(j).Modify
next j
```

# PrtRegion.DeleteItem (Method)

**Syntax:**

**PrtRegion.DeleteItem** *regionLoc, disposition* **[**, *comment$***]**

**Description:**

Deletes an item from the tracking region at a specified region location.

**Comments:**

| Parameter | Description |
|---|---|
| *regionLoc* | Integer. The location of the item in the region. |
| *disposition* | Integer. The disposition of the deleted item. See table below. |
| *comment$* | String. (Optional) - an optional comment to place in the PRT log. |

| Disposition | Description |
|---|---|
| PRT_SCRAP | Scrap the item. |
| PRT_DELETE | Delete the item. |
| PRT_NO_CHECK_SCRAP | Scrap the item overriding region locking status. |
| PRT_NO_CHECK_DELETE | Delete the item overriding region locking status. |

**Example:**

```
Dim Region as new PrtRegion
Region.Id = "PRODUCTION"
Region.GetItemList
'Get all items in a region and delete those items with a
'    specific ItemTypeId.
For i = 0 to Region.ItemCount - 1
   If Region.Item(i).ItemTypeId = "MARKVII" then
      Region.Item(i).DeleteItem i, PRT_DELETE, "Obsolete Type"
   End If
Next i
```

# PrtRegion.GetData (Method)

**Syntax:**

**`PrtRegion.GetData`**

**Description:**

Returns region information such `Status`, `TotalItems`, `Capacity`, `StatusPointId`, and `TotalItemsPointId`, which then become available through the object.

**Example:**

```
'Example of PrtRegion.StatusBit Function using a constant to
'   get the Region Full status bit and display it's status in
'   a message box
Dim DestRegion as new PrtRegion
DestRegion.Id = "Region_Name"              'Replace Region Name
DestRegion.GetData
If DestRegion.StatusBit(PRT_REGION_FULL) Then
       MsgBox "Region Full"
Else
       MsgBox "Region Not Full"
End If
```

# PrtRegion.GetItemList (Method)

**Syntax:**

**`PrtRegion.GetItemList`**

**Description:**

Retrieves a list of items for the current region.

**Comments:**

If you call `PrtRegion.GetItemList` before setting `PrtRegion.Loc` you get the item list for all locations.

To get the item list for a current region location, set PrtRegion.Loc to that location number before calling `PrtRegion.GetItemList`.

If you want to get the item list for all locations and `PrtRegion.Loc` currently contains a location number, set `PrtRegion.Loc` to `PRT_ALL_LOCATIONS` before calling `PrtRegion.GetItemList`.

To get the item list for a particular item class, set **`PrtRegion.ItemClassId`** to the desired item class id before calling **`PrtRegion.GetItemList`**. *See PrtRegion.ItemClassId for more details.*

**Example:**

```
Dim Region as new PrtRegion
Region.Id = "PRODUCTION"
Region.GetItemList
'Get all items in a region and delete those items with a
'    specific ItemTypeId.
For i = 0 to Region.ItemCount - 1
   If Region.Item(i).ItemTypeId = "MARKVII" then
      Region.Item(i).DeleteItem i, PRT_DELETE, "Obsolete Type"
   End If
Next I
```

**See Also**

**`PrtRegion.Loc`**

# PrtRegion.Id (Property Read/Write)

**Syntax:**

**`PrtRegion.Id`**

**Description:**

String property to get or set the Region ID of the object.

**Example:**

```
Dim Region as new PrtRegion
Region.Id = "PRODUCTION"
Region.GetItemList
'Get all items in a region and delete those items with a
'    specific ItemTypeId.
For i = 0 to Region.ItemCount - 1
   If Region.Item(i).ItemTypeId = "MARKVII" then
      Region.Item(i).DeleteItem i, PRT_DELETE, "Obsolete Type"
   End If
Next i
```

# PrtRegion.Item (Function)

**Syntax:**

**`PrtRegion.Item(`***index***`)`**

**Description:**

Returns `PrtItem`. To return a specific item by index from a previous call to `GetItemList`.

**Example:**

```
Dim Region as new PrtRegion
Region.Id = "PRODUCTION"
Region.GetItemList
'Get all items in a region and delete those items with a
'    specific ItemTypeId.
For i = 0 to Region.ItemCount - 1
   If Region.Item(i).ItemTypeId = "MARKVII" then
      Region.Item(i).DeleteItem i, PRT_DELETE, "Obsolete Type"
   End If
Next I
```

# PrtRegion.ItemClassId (Property Read/Write)

**Syntax:**

**`PrtRegion.ItemClassId`**

**Description:**

String property to set or get the Item Class ID in the PrtRegion object. Used in conjunction with the PrtRegion.GetItemList method to retrieve a list of all items of a particular item class in a Tracking Region.

**Example:**

```
Dim Region as new Prtregion
Region.Id = "PRODUCTION"
Region.ItemClassId = “CARRIER”
Region.GetItemList

' Get a list of items with an item class id “CARRIER” from the
'    region “PRODUCTION" and display them one by one in a
'    message box.
Region.ItemClassId = “CARRIER”
Region.GetItemList
for j = 0 to Region.ItemCount - 1
      MsgBox Region.Item(j).ItemId
next j
```

# PrtRegion.ItemCount (Property Read)

**Syntax:**

**`PrtRegion.ItemCount`**

**Description:**

Integer property to get the number of items find by a previous `GetItemList` or `LocateItem` call.

**Example:**

```
Dim Region as new PrtRegion
Region.Id = "PRODUCTION"
Region.GetItemList
'Get all items in a region and delete those items with a
'    specific ItemTypeId.
For i = 0 to Region.ItemCount - 1
   If Region.Item(i).ItemTypeId = "MARKVII" then
      Region.Item(i).DeleteItem i, PRT_DELETE, "Obsolete Type"
   End If
Next i
```

# PrtRegion.ItemType (Function)

**Syntax:**

**PrtRegion.ItemType(***index***)**

**PrtRegion.ItemType(***itemTypeId$***)**

**Description:**

Returns an `PrtItemType` object accessed by a particular index or *itemItemId$*.

The `Region.ItemType(`*index*`)` object supports the properties `.Id` and `.Count`.

**Example:**

```
Dim Region as new PrtRegion
Region.id = "Schedule"
' Display Id and Number for all Item Types in a region
for i = 0 to Region.ItemTypeCount - 1
   x = Region.ItemType(i).Id & " = " & Region.ItemType(i).Count
   msgbox x
next i
```

# PrtRegion.ItemTypeCount (Property Read)

**Syntax:**

**PrtRegion.ItemTypeCount**

**Description:**

Integer property to get the number of Item Type Count records available for the Region. These records provide totals of item Type IDs for a region.

**Example:**

```
Dim Region as new PrtRegion
Region.id = "Schedule"
' Display Id and Number for all Item Types in a region
for i = 0 to Region.ItemTypeCount - 1
   x = Region.ItemType(i).Id & " = " & Region.ItemType(i).Count
   msgbox x
next i
```

# PrtRegion.LocateItem (Method)

**Syntax:**

**`PrtRegion.LocateItem`** *itemId$, refId$* **[**, *numRetries***]**

**Description:**

Locates a serialized item within the tracking region. Upon return, `PrtRegion.ItemCount` contains the number of items found, and may access each found item through the `PrtRegion.Item` function.

**Comments:**

| Parameter | Description |
|---|---|
| *itemId$* | String. The Item ID to locate, may be an empty string if *refId$* is specified. |
| *refId$* | String. The Reference ID of the item to locate, may omitted or an empty string if *itemId$* is specified. |
| *numRetries* | Integer (optional) - The number of times to retry to find the item before failing. If an item is in transition from one tracking region to the next, there is a small period of time when it doesn't appear in either regions tracking queue. Defaults to 3 if not specified. |

**Example:**

```
Dim Region as new PrtRegion
Region.Id = "SCHEDULE"
' Get a single item from a region with its RefId and display
'    its ItemId in a message box.
Region.LocateItem "",REFNUM$
MsgBox Region.Item(0).ItemId
```

# PrtRegion.Loc (Property Read/Write)

**Syntax:**

**`PrtRegion.Loc`**

**Description:**

Integer property to set a specific region location for a subsequent `GetItemList`. The search will then only return a list of items which exist in that single location.

**Example:**

```
Dim Region as new PrtRegion
' Get a single item from a region with its RefId and display
'    its ItemId in a message box.
Region.Loc = 0
Region.GetItemList
'Get the Item list from the first location in the region and
'    display the results in a message box
If Region.ItemCount = 0 then
       MsgBox "Region " & Region & " has no Items."
Else
       Msgbox "Found " & Region.ItemCount & ' items."
End If
```

# PrtRegion.MaxLocation (Property Read)

**Syntax:**

**`PrtRegion.MaxLocation`**

**Description:**

Integer property to get the last occupied region location.

**Example:**

```
Dim Region as new PrtRegion
Region.Id = "PRODUCTION"
Region.GetData   'Get current max. location information
Region.Loc = Region.MaxLocation
'Get all items in the last occupied region location and
' display them one by one in a message box.
Region.GetItemList
for j = 0 to Region.ItemCount - 1
      MsgBox Region.Item(j).ItemId
next j
```

# PrtRegion.Modify (Method)

**Syntax:**

**`PrtRegion.Modify [`***comment$***`] [`***, userId$***`]`**

**Description:**

Sends the modified status mask to PRT.

**Comments:**

| Parameter | Description |
|---|---|
| *comment$* | String. Optional. A comment to be written to the PRT log file. |
| *userId$* | String. Optional. The User ID to be written to the PRT log file. |

**Example:**

```
Dim Prt as new Prt
n% = 17
' Get a list of items from PRT and initialize a particular
'    status bit for each one.
Prt.GetRegionList
for j = 0 to prt.ItemCount - 1
       Prt.Region(j).ClearStatusBit n%
       Prt.Region(j).Modify
next j
```

# PrtRegion.ProjectId (Property Read/Write)

**Syntax:**

**`PrtRegion.ProjectId`**

**Description:**

String property to set the project that subsequent Production Tracking extensions will communicate with.

**Example:**

```
dim prt as new PrtRegion
' Get a list of services from PRT and display them one by one
'    in a message box
prt.ProjectId = "\\project1"
prt.GetItemList
for j = 0 to prt.ItemCount - 1
       MsgBox prt.Item(j).ItemId
next j
```

# PrtRegion.Reorder (Method)

**Syntax:**

**`PrtRegion.Reorder`** *reorderArray* **`[`**, *comment$* **`[`**, *userId$***`]]`**

**Description:**

Reorders the locations within a tracking region. This has the effect of moving all the items at each location in the region to another (possibly the same) location in the region.

**Comments:**

| Parameter | Description |
|---|---|
| *reorderArray* | Array, integer. An integer array consisting of region locations.<br><br>These location numbers (1 to **`num_locations`**), specify the new region location for items residing in their current location, that is the locations the items are to be moved to. The *reorderArray* array index implicitly define the current location number,<br><br>For example, the first element in the array (*reorderArray*[0]) refers to the items that currently reside in region location 1. The value of *reorderArray[*0] specifies the region location where the items currently in region location 1 are to be moved to.<br><br>This array must include elements for each location currently occupied in the region. |
| *comment$* | String. (Optional) A comment to be recorded into the PRT history log file. |
| *UserId$* | String. (Optional) This is used for logging purposes and is optional. If not supplied the host process name is used. |

**Example:**

```
Dim Region as new PrtRegion
Region.id = "OUTPUT"
'Invert the items in a region from first to last
Dim Array(99) as integer
For x% = 0 to 99
       Array(x%) = 100 - x%
Next x%
Region.Reorder Array
```

# PrtRegion.SetPoints (Method)

**Syntax:**

**`PrtRegion.SetPoints`** *enable*

**Description:**

Enables or disables the item transition setpoints configured in the PRT tracking route records. These points are set by PRT when an Item transitions into the configured destination region.

| | |
|---|---|
| *enable* | Boolean (TRUE or FALSE) |

**Example:**

```
Dim Region as new PrtRegion
Region.Id = "PRODUCTION"
If TIME > 18:00:00 or TIME < 6:00:00 then
       Region.SetPoints False
Else
       Region.SetPoints True
End If
```

# PrtRegion.SetStatusBit (Method)

**Syntax:**

**`PrtRegion.SetStatusBit`** *b%*

**Description:**

Clears a particular status bit in the regions status mask.

**Comments:**

To obtain the region's status mask, use the `PrtRegion.GetData` method. To update the Region's status mask to PRT use the `PrtRegion.Modify`.

Bits 0-15 are for User Use. Bits 16-31 are for GE Fanuc use. The following constants may be used to access GE Fanuc specific bits.

| Constants | Description |
|---|---|
| PRT_REGION_FULL | Region Full Bit |
| PRT_REGION_EXT_HOLD | Region External Hold Bit |
| PRT_REGION_IN_LOCK | Region in Lock Bit |
| PRT_REGION_OUT_LOCK | Region Out Lock Bit |
| PRT_REGION_OUT_OF_SEQ | Region Out of Seq Bit |
| PRT_REGION_CAP_EXCEEDED | Region Capacity Exceeded Bit |
| PRT_REGION_NORMAL | Region Normal Bit |

**Example:**

```
Dim Prt as new prt
Prt.GetRegionList
'Get a list of regions in PRT, check each one’s first status
'    bit, then set it if it’s not already set.
For J% = 0 to Prt.RegionCount -1
       If Prt.Region(J%).StatusBit(0) = FALSE Then
              CurPrt.Item(I).SetStatusBit 0
              CurPrt.Item(I).Modify
End If
next J%
```

# PrtRegion.Status (Property Read/Write)

**Syntax:**

**PrtRegion.Status**

**Description:**

Integer property to get or set the region's status mask.

**Example:**

```
Dim Region as new PrtRegion
Region.Id = "PRODUCTION"
Region.GetData
'Perform a quick check for any status bits for a region being
'    set and display the results in a message box.
If Region.Status > 0 then
       msgbox "One or more status bits are set."
Else
       msgbox "No status bits set."
End If
```

# PrtRegion.StatusBit (Method)

**Syntax:**

**`PrtRegion.StatusBit(`***b%***`)`**

**Description:**

Accesses a particular status bit in the region's status mask.

**Comments:**

To obtain the region's status mask from PRT, use the `PrtRegion.GetData` method.

Bits 0-15 are for User Use. Bits 16-31 are for GE Fanuc use. The following constants may be used to access GE Fanuc specific bits.

| Constants | Description |
|---|---|
| PRT_REGION_FULL | Region Full Bit |
| PRT_REGION_EXT_HOLD | Region External Hold Bit |
| PRT_REGION_IN_LOCK | Region in Lock Bit |
| PRT_REGION_OUT_LOCK | Region Out Lock Bit |
| PRT_REGION_OUT_OF_SEQ | Region Out of Sequence Bit |
| PRT_REGION_CAP_EXCEEDED | Region Capacity Exceeded Bit |
| PRT_REGION_NORMAL | Region Normal Bit |

**Example:**

```
'Example of PrtRegion.StatusBit Function using a constant to
'   get the Region Full status bit and display it’s status in'
    a message box
Dim DestRegion as new PrtRegion
DestRegion.Id = "Region_Name"              'Replace Region Name
DestRegion.GetData
If DestRegion.StatusBit(PRT_REGION_FULL) Then
       MsgBox "Region Full"
Else
       MsgBox "Region Not Full"
End If
```

# PrtRegion.StatusPointId (Property Read)

**Syntax:**

**PrtRegion.StatusPointId**

**Description:**

String property to get the Point ID configured to contain the status point for the region or an empty string if the status point is not configured for the region.

**Example:**

```
Dim Region as new PrtRegion
Region.Id = "PRODUCTION"
'Get the name of a region's status point and display its
'   contents in a message box.
StatPtId$ = Region.StatusPointId
If StatPtId$ <> "" then
       msgbox "Status: " & PointGet(StatPtId$)
Else
       msgbox "No point configured!"
End If
```

# PrtRegion.TotalItems (Property Read)

**Syntax:**

**PrtRegion.TotalItems**

**Description:**

Integer property to get the total number of items in the region.

**Example:**

```
Dim Region as new PrtRegion
Region.Id = "PRODUCTION"
'Get a region's data and display the total number of items it
'   contains in a message box.
Region.GetData
msgbox "Region "& Region.id &" contains "& Region.TotalItems &"
      items."
```

# PrtRegion.TotalItemsPointId (Property Read/Write)

**Syntax:**

**`PrtRegion.TotalItemsPointId`**

**Description:**

String property to get the Point ID configured to contain the total number of items in the region or an empty string if the ID is not configured.

**Example:**

```
Dim Region as new PrtRegion
Region.Id = "PRODUCTION"
'Get the name of a region's Total Item point and display its
'   contents in a message box.
TotPtId$ = Region.TotalItemsPointId
If TotPtId$ <> "" then
       msgbox "Total Items: " & PointGet(TotPtId$)
Else
       msgbox "No point configured!"
End If
```

# PrtService.GetGroupList (Method)

**Syntax:**

**`PrtService.GetGroupList`**

**Description:**

Gets a list of groups contained in the PRT Service. Upon return, the total group count may be obtained from `GroupCount`, and the individual groups may be accessed via the `Group` function.

**Example:**

```
Dim ServiceA as new PrtService
ServiceA.Id = "SERVICE_A"
' Get a list of groups in the service and display them one by
'   one in a message box.
ServiceA.GetGroupList
for j = 0 to ServiceA.GroupCount - 1
      MsgBox ServiceA.Group(j).Id
next j
```

# PrtService.GetItemList (Method)

**Syntax:**

**`PrtService.GetItemList`**

**Description:**

Gets a list of all items for the tracking service. Upon retrun, the total item count may be obtained from `ItemCount` and the individual items may be accessed via the `Item` function.

To get the item list for a particular item class, set `PrtService.ItemClassId` to the desired item class id before calling `PrtService.GetItemList`. *See PrtService.ItemClassId for more details.*

**Example:**

```
Dim ServiceA as new PrtService
ServiceA.Id = "SERVICE_A"
' Get a list of items in the service and display them one by
'   one in a message box.
ServiceA.GetItemList
for j = 0 to ServiceA.ItemCount - 1
      MsgBox ServiceA.Item(j).Id
next j
```

# PrtService.GetRegionList (Method)

**Syntax:**

**`PrtService.GetRegionList`**

**Description:**

Gets a list of Tracking regions contained in the PRT Service. Upon return, the total region count may be obtained from `RegionCount`, and the individual regions may be accessed via the `Region` function.

**Example:**

```
Dim ServiceA as new PrtService
ServiceA.Id = "SERVICE_A"
' Get a list of regions in the service and display them one by
'   one in a message box.
ServiceA.GetRegionList
for j = 0 to ServiceA.RegionCount - 1
      MsgBox ServiceA.Region(j).Id
next j
```

# PrtService.Group (Function)

**Syntax:**

**`PrtService.Group(`***index*`)`**

**Description:**

Returns a group at a specified index from the list which was found by a previous call to `GetGroupList`.

**Example:**

```
Dim ServiceA as new PrtService
ServiceA.Id = "SERVICE_A"
' Get a list of groups in the service and display them one by
'   one in a message box.
ServiceA.GetGroupList
for j = 0 to ServiceA.GroupCount - 1
      MsgBox ServiceA.Group(j).Id
next j
```

# PrtService.GroupCount (Property Read)

**Syntax:**

**`PrtService.GroupCount`**

**Description:**

Integer property to get the number of Groups found by a prior call to `GetGroupList`.

**Example:**

```
Dim ServiceA as new PrtService
ServiceA.Id = "SERVICE_A"
' Get a list of groups in the service and display them one by
'   one in a message box.
ServiceA.GetGroupList
for j = 0 to ServiceA.GroupCount - 1
      MsgBox ServiceA.Group(j).Id
next j
```

# PrtService.Id (Property Read/Write)

**Syntax:**

**PrtService.Id**

**Description:**

String property to get or set the Service ID associated with the object.

**Example:**

```
Dim ServiceA as new PrtService
ServiceA.Id = "SERVICEA_DC"
' Get a list of groups in the service and display them one by
'   one in a message box.
ServiceA.GetGroupList
for j = 0 to ServiceA.GroupCount - 1
      MsgBox ServiceA.Group(j).Id
next j
```

# PrtService.Item (Function)

**Syntax:**

**PrtService.Item(***index***)**

**Description:**

Returns an Item at the specified index from the list which was found by a previous call to `GetItemList`.

**Example:**

```
Dim ServiceA as new PrtService
ServiceA.Id = "SERVICEA_DC"
' Get a list of items in the service and display them one by
'   one in a message box.
ServiceA.GetItemList
for j = 0 to ServiceA.ItemCount - 1
      MsgBox ServiceA.Item(j).Id
next j
```

# PrtService.ItemClassId (Property Read/Write)

**Syntax:**

**`PrtService.ItemClassId`**

**Description:**

String property to set or get the Item Class ID in the PrtService object. Used in conjunction with the PrtService.GetItemList method to retrieve a list of all items of a particular item class in a Tracking Service.

**Example:**

```
Dim ServiceA as new PrtService
ServiceA.Id = "SERVICE_A"
ServiceA.ItemClassId = “CARRIER”
' Get a list of items with an item class id “CARRIER” from the
'   service “SERVICE_A” and display them one by one in a
'   message box.
ServiceA.GetItemList
for j = 0 to ServiceA.ItemCount - 1
      MsgBox ServiceA.Item(j).Id
next j
```

# PrtService.ItemCount (Property Read)

**Syntax:**

**`PrtService.ItemCount`**

**Description:**

Integer property to get the number of items find by a previous `GetItemList` or `LocateItem` call.

**Example:**

```
Dim ServiceA as new PrtService
ServiceA.Id = "SERVICEA_DC"
' Get a list of items in the service and display them one by
'   one in a message box.
ServiceA.GetItemList
for j = 0 to ServiceA.ItemCount - 1
      MsgBox ServiceA.Item(j).Id
next j
```

# PrtService.LocateItem (Method)

**Syntax:**

**PrtService.LocateItem** *itemId, refId* **[**, *numRetries***]**

**Description:**

Locates a serialized item within the tracking service. Upon return, the total item count may be obtained from `ItemCount`, and the individual items may be accessed via the `Item` function.

**Comments:**

| Parameter | Description |
|---|---|
| *itemId$* | String. The Item ID to locate, may be an empty string if *refId$* is specified. |
| *refId$* | String. The Reference ID of the item to locate, may omitted or an empty string if *itemId$* is specified. |
| *NumRetries* | Integer (optional) - The number of times to retry to find the item before failing. If an item is in transition from one tracking region to the next, there is a small period of time when it doesn't appear in either regions tracking queue. Defaults to 3 if not specified. |

**Example:**

```
Dim ServiceA as new PrtService
ServiceA.Id = "SERVICEA_DC"
' Get a single item from a region with its RefId and display
'   its ItemId in a message box.
ServiceA.LocateItem "",REFNUM$
MsgBox ServiceA.Item(0).ItemId
```

# PrtService.Region (Function)

**Syntax:**

**PrtService.Region(***index***)**

**Description:**

Returns a region at the specified index from the list which was found by a previous call to `GetRegionList`.

**Example:**

```
Dim ServiceA as new PrtService
ServiceA.Id = "SERVICEA_DC"
' Get a list of regions in the service and display them one by
'   one in a message box.
ServiceA.GetRegionList
for j = 0 to ServiceA.RegionCount - 1
      MsgBox ServiceA.Region(j).Id
next j
```

# PrtService.RegionCount (Property Read)

**Syntax:**

**`PrtService.RegionCount`**

**Description:**

Integer property to get a count of the number of Tracking Regions found by a prior `GetRegionList` call.

**Example:**

```
Dim ServiceA as new PrtService
ServiceA.Id = "SERVICEA_DC"
' Get a list of regions in the service and display them one by
'   one in a message box.
ServiceA.GetRegionList
for j = 0 to ServiceA.RegionCount - 1
      MsgBox ServiceA.Region(j).Id
next j
```

# Appendix F - Configurable Tracking Interface

## About the Configurable Tracking Interface

This appendix provides the information necessary to configure and use the Configurable Tracking Interface (CTI). CTI lets you perform certain Tracker functions through CIMPLICITY Points, rather than through API/BCE calls or the User Interface.

CTI can:

- Add an Item to Tracker
- Modify an existing Item
- Move an Item
- Confirm an Item exits in the system.

## CIMPLICITY Environment

The Configurable Tracking Interface application software is compatible with CIMPLICITY NT release 1.0 or later. The software was constructed under Windows NT 3.5 Release Candidate 1, using the following:

- CIMPLICITY Point Management API
- CIMPLICITY Production Tracking API

The Configurable Tracking Interface executable exists in **`%BSM_ROOT%\EXE\`** on the NT platform for access by run-time CIMPLICITY. The name of the executable is **`CTI_RP.EXE`**.

**Note:** You may need access to the documentation for the APIs listed above.

# CIMPLICITY Base System Configuration

It is assumed that you will reference the appropriate base system documentation to configure the points to be utilized by the CTI software.

- All CTI input and response points must be configured as either text or analog array points.
- All CTI Transition failure points must be configured as a single element digital point.

# Site Configuration

The CTI application must be configured using a text editor directly in the following process configuration files:

- SERVICE.IDT
- LOGPROC.IDT
- NODE_LOGPROC.IDT
- PHYSPROC.IDT
- MASTER.MCP

### *SERVICE.IDT*

The fields of each service data record are:

- Service Identifier
- Subsystem Identifier
- Process Identifier

An example of the data records in the SERVICE.IDT file follows:

```
CTI_RP_1|CTI|CTI_RP_1
CTI_RP_2|CTI|CTI_RP_2
```

## LOGPROC.IDT

The fields of each data record are:

- Process Identifier
- Process Type
- PM Flags
- Description

The final field in each record of this file is a description of the process. This field may be filled in as the user chooses.

An example of these data records follows:

```
CTI_RP_1|RESIDENT|1|Configurable Tracking Interface Data Monitor
CTI_RP_2|RESIDENT|1|Configurable Tracking Interface Data Monitor
```

## NODE_LOGPROC.IDT

For CIMPLICITY HMI for Windows NT, only the first three fields of this file are used. Use the standard values for other processes in the CTI records. The first field indicates the node that the CTI Resident process will run on. The second field in the file is the Process ID of the process, as configured in the **`logproc.idt`** file. The third field is the name of the executable to be run. This is typically **`BSM_ROOT:\exe\cti_rp.exe`**, unless you have placed the executable in a different location.

An example of these data records follows:

```
NODE_1|CTI_RP_1|BSM_ROOT:\exe\alpha\cti_rp.exe|4|1|1|-
80|56|50000|0|50|2000|100|50000|10|50|300|3072|768|-1
NODE_1|CTI_RP_2|BSM_ROOT:\exe\alpha\cti_rp.exe|4|1|1|-
80|56|50000|0|50|2000|100|50000|10|50|300|3072|768|-1
```

## PHYSPROC.IDT

The Process ID and the object name are the same for ease of use. Since the object name has a ten-character restriction, the process id is ten characters as well as the Subsystem ID and the Service ID.

The fields of each data record are:

- Node name
- Process Identifier
- Object Name
- Redundant or allocated process order (always 0)

An example of the entries for the CTI subsystem follows:

```
NODE_1|CTI_RP_1|CTI_RP|0
NODE_1|CTI_RP_2|CTI_RP|0
```

### *MASTER.MCP*

This is a text-only configuration file, which does not need to be converted using SCPOP. It contains the list of processes to be started automatically when the CIMPLICITY project starts. If you wish CTI to start automatically, simply list the name of the CTI process you have configured in the previous four files as the last line of this file.

An example MASTER.MCP file follows:

```
MASTER
UR_RP
AMRP
MASTER_PTM0_RP
PTX_RP
DL_RP
DYNCFG
MASTER_PTDP_RP
EM_RP
PTDL_RP
APRT_DC
BPRT_DC
APRT_DS
BPRT_DS
PRTGRD
PRT_STND
PDC_DS
CTI_RP_1
CTI_RP_2
```

# Configurable Tracking Interface-Specific Configuration Files

The CTI application software has multiple configuration files that must be maintained separately from CIMPLICITY configuration. These files are:

- CTI_RP.CFG
- CTI_RP_LAYOUT.CFG
- CTI_RP_XLATE.CFG
- CTI_PT_DEF.CFG
- PRT_SRV_ATTR.CFG

These files reside in the directory **`%SITE_ROOT%\DATA`** for CIMPLICITY HMI for Windows NT, and are manually edited. The files have an IDT-file format; hence, the first record containing "**`|-*`**" indicates that "**`|`**" is the field delimiter character, "**`-`**" is the line continuation character, and "**`*`**" is the line comment character.

The correct names for the first three files are determined by replacing the sequence "CTI_RP" in each file name with the name of the CTI resident process configured in the previous section. The filename CTI_PT_DEF.CFG is constant. PRT_SRV_ATTR.CFG's file name is determined by replacing "PRT_SRV" with the resident process name for the PRT Data Server which CTI will monitor (note that this means a file name of "_ATTR.CFG" in the case that CTI will monitor multiple servers) .

The PRT field names used in all CTI configuration files will either be an attribute name as defined in PRT, or one of the following constant field names:

- PRT_ITEM_ID
- PRT_REFERENCE_ID
- PRT_ITEM_STATUS
- PRT_ITEM_TYPE_ID

If PRT_ITEM_STATUS is used, an optional bit offset can be specified using the following format:

```
PRT_ITEM_STATUS:n
```

where n is the bit offset. 0 is the rightmost (least significant bit). If a bit offset is specified, only the particular bit of the item status is changed based on the point value. The other bits in the status word remain the same. If no bit offset is specified, then the item status is changed to be the point value received.

## CTI / PRT Mapping File (CTI_RP.CFG)

This file contains a single field, the name of the Resident Process for the PRT data server that CTI will monitor, without the "_DS" extension. In the case that you wish to monitor all data servers, leave this field blank.

The following are examples of the two possibilities:

- For one data server:

  ```
  |-*
  APRT
  ```

- For all data servers:

  ```
  |-*
  ```

## Field Layout File (CTI_RP_LAYOUT.CFG)

The configuration file CTI_LAYOUT.CFG defines the layout of specific PRT fields within each point received by CTI. The records in the file contain the following fields:

| | |
|---|---|
| RECEIVED POINT ID | The name of the CIMPLICITY TEXT or ANALOG point that will be used by CTI to receive data. |
| ORDER | The order in which the PRT fields are stored within the overall text point. Values must begin at 1, must be unique, and must be monotonically increasing.<br><br>Field #1 is the field that CTI will use to search the PRT database, and should be either PRT_ITEM_ID or PRT_REFERENCE_ID. The remaining fields are data passed to CTI for updating the Item record once it is found. |
| FIELD ID | The ID of a PRT field or attribute to be modified. This will be the same attribute identifier that is used by PRT. |
| FIELD LENGTH | For TEXT points, the length in characters of the field and attribute data. |

Examples of records found in the **CTI_LAYOUT.CFG** file are:

```
|-*
RCV_POINT_1|1|PRT_ITEM_ID|20
RCV_POINT_1|2|ATT_1|4
RCV_POINT_1|3|ATT_3|6
RCV_POINT_1|4|ATT_5|4
RCV_POINT_1|5|PRT_ITEM_STATUS:1|1
RCV_POINT_1|6|PRT_ITEM_STATUS:4|1
RCV_POINT_2|1|PRT_REFERENCE_ID|
RCV_POINT_2|2|ATT_1|
RCV_POINT_2|3|ATT_7|
RCV_POINT_2|4|ATT_4|
RCV_POINT_2|5|PRT_ITEM_STATUS|
```

The above records represent the layout of PRT field data within the points used by CTI to receive data. An example of what the record stands for follows:

The point RCV_POINT_1 contains data for the ITEM_ID field in the PRT record, the attributes ATT_1, ATT_3 and ATT_5, and item status bit 1 and item status bit 4 in that order.

Since **RCV_POINT_1** is a text point, the length of each attribute must be specified. The length of the ITEM_ID data is 20 bytes, the length of the ATT_1 data is 4 bytes, the length of the ATT_3 data is 6 bytes and the length of the ATT_5 data is 4 bytes. The length of **PRT_ITEM_STATUS:1** and **PRT_ITEM_STATUS**:4 are each 1 byte.

The point **RCV_POINT_2** contains data for the item ID, attributes ATT_1, ATT_7 and ATT_4 and item status in that order.

Since **RCV_POINT_2** is an analog point, the length field will be left empty.

## Field Translation File (CTI_XLATE.CFG)

The configuration file **CTI_XLATE.CFG** will define the translation of integer field values to ASCII text values by CTI. ANALOG array points received by CTI may have a translation configured for each of the field values stored in the point. The CTI will convert the integer value to an ASCII text value if such a translation is configured in this file. If no translation is configured, the integer field value received will be treated as the ASCII character for the same value. The records in this file contain the following fields:

| | |
|---|---|
| RECEIVED POINT ID | The name of the CIMPLICITY ANALOG array point that will be used by CTI to receive data. |
| FIELD ID | The ID of a PRT field whose integer value will be translated by CTI.<br><br>This will be the same field identifier that is used in the file **PRT_API.H** in the structure PRT_ITEM_STRUCT for that field. |
| NUMERIC VALUE | The numeric value that will be translated to ASCII text by CTI. |
| TEXT VALUE | The ASCII text value that corresponds to the above numeric value. |

Examples of records found in the **CTI_XLATE.CFG** file are:

```
|-*
RCV_POINT_2|ATT_1|0|TEXT_VALUE_1
RCV_POINT_2|ATT_1|1|TEXT_VALUE_2
RCV_POINT_2|ATT_1|2|TEXT_VALUE_3
RCV_POINT_2|ATT_1|3|TEXT_VALUE_4
RCV_POINT_2|ATT_7|100|OTHER_TEXT_1
RCV_POINT_2|ATT_7|101|OTHER_TEXT_2
RCV_POINT_2|ATT_7|102|OTHER_TEXT_3
```

The above records represent the translation that will be performed for various attribute values when they are obtained by CTI from an ANALOG array point. Attributes which do not have any translation configured will be converted by CTI into the ASCII character equivalent for the same value. An example of what the record stands for follows:

The point **RCV_POINT_2** has an attribute-field relationship defined in **CTI_LAYOUT.CFG**.

If the field representing the attribute ATT_1 contains the integer value zero, CTI will assign the ASCII text value TEXT_VALUE_1 to the corresponding attribute in the PRT data record when an update is performed.

Since the attributes ATT_3, ATT_4 and ATT_5 and the item status are not specified for this point, CTI will translate the integer values for these attributes into their ASCII character equivalents.

## Point Definition File (CTI_PT_DEF.CFG)

The configuration file **CTI_PT_DEF.CFG** defines the points, which the CTI will use to send and receive data. The records in the file contain the following fields:

| | |
|---|---|
| RECEIVED POINT ID | The name of the CIMPLICITY TEXT or ANALOG point that will be used by CTI to receive data. |
| CTI PROCESS ID | The process ID, as configured in the system configuration files, of the CTI process that will handle modification requests through the received point. |
| RESPONSE POINT ID | The name of the CIMPLICITY TEXT or ANALOG point that will be used by CTI to send its response. |
| PRT FIELD NAME | The name of a field in the PRT record that will be sent as part of the response (typically this is the item ID).<br><br>This must be one of the fields that is defined in the CTI_LAYOUT.CFG file for this point. The value returned will be that sent to CTI via the received point, not one returned from tracking.<br><br>If this field is left blank, the response point will not contain any field value. |
| SUCCESS VALUE | A value sent as part of the response that will be used to indicate a successful transaction. |
| FAILURE VALUE | A value sent as part of the response that will be used to indicate an unsuccessful transaction. |
| DEST REGION | If the item is not found in PRT inventory, this region is the region that the item will be added to if adds are allowed for this point.<br><br>If the item already exists in inventory, this will be the region that the item will be moved to after it has been modified, if moves are allowed. |

| | |
|---|---|
| PRIOR REGION | The PRT region that an item will be moved from when it is modified.<br><br>If a region is specified, the item must be at the head of this region for an item to be updated and moved to the destination region.<br><br>If no region is specified, the item will always be moved to the destination region, if moves are allowed. |
| TRANSITION FLAG | Flag indicating what types of transitions are allowed. This flag indicates if new items can be added to the destination region and if existing items should be moved to the destination region when updated.<br><br>1 = Only adds are allowed<br><br>2 = Only moves are allowed<br><br>3 = Adds and moves are allowed<br><br>Other = Adds and moves NOT allowed |
| ITEM TYPE | A value representing the type of item that may be created when the New Item Allowed field is set to 1.<br><br>If this field is left blank, any item type may be created. |
| TRANSITION FAILURE POINT | The name of the CIMPLICITY Digital point that is used to represent the success or failure of an item moving into (or being added to) the configured PRT Destination Region.<br><br>The point will be set to 1 if the move is successful, and 0 if the move is unsuccessful. |
| IGNORE FIRST VALUE | A value that specifies whether or not CTI will ignore the first point update received for an item.<br><br>A value of 1 specifies that the CTI will ignore the first value received for a point.<br><br>A value of 0 specifies that CTI will respond to the first value received for a point.<br><br>An ANALOG response point must have integer values configured for both the SUCCESS VALUE and FAILURE VALUE fields.<br><br>A TEXT response point will have alphanumeric values configured for the SUCCESS VALUE and FAILURE VALUE fields. |

Examples of records found in the CTI_PT_DEF.CFG file are:

```
|-*
RCV_POINT_1|CTI_RP|RSP_POINT_1|PRT_ITEM_ID|SUCCESS1|-
FAILURE1|R1_BOX||3||TRANS_PT_1|1
RCV_POINT_2|CTI_RP|RSP_POINT_2|PRT_REFERENCE_ID|-
      SUCCESS2|FAILURE2|R1_BOX||1|25||1
RCV_POINT_3|CTI_RP|RSP_POINT_3|PRT_ITEM_ID|SUCCESS3|-
FAILURE3|R1_BOX||2|||1
RCV_POINT_4|CTI_2_RP|RSP_POINT_4||SUCCESS4|FAILURE4|-
R1_BOX|R6_DROP_BOX|2|||1
RCV_POINT_5|CTI_RP|RSP_POINT_5|PRT_ITEM_ID|1|0|||||1
RCV_POINT_6|CTI_RP|RSP_POINT_6|PRT_REFERENCE_ID|1|0-
||||||1
```

The above records correspond to points used by CTI to receive data and send responses. An example of what the records stand for follows:

- The point **RCV_POINT_1** will be used by the CTI process CTI_RP to receive updated PRT data.
- The point **RSP_POINT_1** will be used to send responses. The response point will contain the value of the PRT_ITEM_ID field as well as either the text value "SUCCESS1" for a successful transaction or the text value FAILURE1 for an unsuccessful transaction.
- Items not found in inventory will be added to the R1_BOX region. Existing items that are not already in the R1_BOX region will be moved to the R1_BOX region.
- If either an add or a move fails, **TRANS_PT_1** will be set to 1. It will be up to an operator to reset this point to 0.
- The first response received when CTI_RP is started and when **RCV_POINT_1** transitions from unavailable to available will be ignored.
- The point **RCV_POINT_2** will be used by the CTI process CTI_RP to receive updated PRT data.
- The point **RSP_POINT_2** will be used to send responses. The response point will contain the value of the PRT_REFERENCE_ID field as well as either the text value SUCCESS2 for a successful transaction or the text value FAILURE2 for an unsuccessful transaction.
- Items not found in inventory which have an item type of 25 will be added to the R1_BOX region. No moves are allowed on existing items.
- The first response received when CTI_RP is started and when **RCV_POINT_2** transitions from unavailable to available will be ignored.
- The point **RCV_POINT_3** will be used by the CTI process CTI_RP to receive updated PRT data.
- The point **RSP_POINT_3** will be used to send responses. The response point will contain the value of the PRT_ITEM_ID field as well as either the text value SUCCESS3 for a successful transaction or the text value FAILURE3 for an unsuccessful transaction.
- Items not found in inventory will NOT be added. Existing items that are not already in the R1_BOX region will be moved to the R1_BOX region.
- The first response received when CTI_RP is started and when **RCV_POINT_3** transitions from unavailable to available will be ignored.

- The point **RCV_POINT_4** will be used by the CTI process CTI_2_RP to receive updated PRT data.
- The point **RSP_POINT_4** will be used to send responses. The response point will contain either the text value SUCCESS4 for a successful transaction or the text value FAILURE4 for an unsuccessful transaction.
- Items not found in inventory will NOT be added. Existing items will be updated and moved to the R1_BOX region only if they are at the head of the 6_DROP_BOX region.
- The first response received when CTI_2_RP is started and when **RCV_POINT_4** transitions from unavailable to available will be ignored.

## Attribute Template File (PRT_SRV_ATTR.CFG)

The configuration file **CTI_ATTRIBUTES.CFG** defines the attribute templates that CTI will use when updating PRT for each item type. The records in the file will contain a PRT Item Type ID followed by the attribute names to be configured for that item type.

An empty item type field indicates that the record should be used as the default attribute template if no record is defined for an item type (Note that all attributes listed in a default template will be created/updated for ANY item type not otherwise defined in this file, replacing existing attribute values, and that this will occur whenever CTI is asked to locate an item, regardless of whether the input point is configured to input values.).

When an item is updated by CTI, CTI will create an attribute record for each attribute in the template. If a specified attribute already exists for the item or is defined by the point update, the value will be stored; otherwise the attribute value will be set to NULL. Any existing attributes that are not defined in the template will be maintained, but placed at the end of the list of attributes.

If no attribute template is defined for an item type and no default attribute template is configured, the existing attributes will be maintained.

Examples of records found in the CTI_ATTRIBUTES.CFG file are:

```
|-*
25|CONFIG CODE|UPPER COLOR|PS SEQ #|APN CODE|-
GA SEQ #|HOLD CODE|FIRST RUN|ASRS BIN|DEST
|CONFIG CODE|UPPER COLOR|PS SEQ #|APN CODE|-
GA SEQ #|HOLD CODE|FIRST RUN|ASRS BIN|DEST
```

The above records represent the attribute template that will be used by the CTI for different PRT item types. The first template will be used for PRT item type 25; all other PRT item types will use the second template.

For example, an existing item of item type 25 is updated by CTI. Its current attributes are the following:

| Name | Value |
|---|---|
| PS SEQ # | 666666 |
| INSPECTION CODE | 12321111 |

A point is received which sets the PS SEQ # to 666534, and CONFIG CODE to 0Y*. When the point is done being updated, the attributes for the item will be the following:

| Name | Value |
|---|---|
| CONFIG CODE | 0Y* |
| UPPER COLOR | |
| PS SEQ # | 666534 |
| APN CODE | |
| GA SEQ# | |
| HOLD CODE | |
| FIRST RUN | |
| ASRS BIN | |
| DEST | |
| INSPECTION CODE | 12321111 |

# CTI Disable Logical

The logical CTI_DISABLE controls the processing of points received by the CTI. If CTI_DISABLE is set to Yes then CTI will take no action when it receives points. If CTI_DISABLE is undefined or set to a value other than Yes, CTI will process points and perform PRT modifications normally. The CTI_DISABLE logical is referenced by CTI whenever a point update is received, so CTI does not need to be bounced for the logical to take effect.

# Layout of Data Points

The CIMPLICITY points used to receive data are composed of a database key followed by the data for one or more PRT record fields to be modified. All PRT attributes are modifiable as well as the following PRT fields:

- Item Status–Numeric field
- Item ID–Alphanumeric field
- Reference ID–Alphanumeric field
- Item Type ID–Alphanumeric field

Any PRT record field or attribute received by a CTI process may be used in the response point as a response field.

The CIMPLICITY TEXT points used to receive data must contain the following fields when populated:

| | |
|---|---|
| PRT Database key | The first 20 bytes of the text point will contain a key into the PRT database (either the Item ID or the Reference ID of the item to be modified).<br><br>The key must be identical to the field value used by PRT for that item. If the database key is less than 20 bytes, the remaining bytes will be padded with spaces. |
| PRT Field Data | The data for each PRT field to be modified will immediately follow and must be in the same format and order that is configured in the file **CTI_LAYOUT.CFG**.<br><br>The field data items must be contiguous and blank padding will be used when necessary. |

The CIMPLICITY ANALOG array points used to receive data must contain the following fields when populated:

| | |
|---|---|
| PRT Database Key | The first element of the array point must contain a value representing the numeric field value to be used as a database key (usually the Item ID or Reference ID of the item to be modified).<br><br>This integer value will be translated into an ASCII character string equivalent by CTI. For example, the integer value 4731 would be translated to the ASCII value 4731. |
| PRT Field Data | The remaining elements of the array point must contain values representing each PRT field to be modified in the order that they are configured in the file **CTI_LAYOUT.CFG**. |

The CTI may translate these integer values into ASCII text values if this is defined in the file **CTI_XLATE.CFG**.

The CIMPLICITY TEXT points used to send a response must contain the following fields when populated:

| | |
|---|---|
| PRT Record Field | The first field of the text response point will contain the value of the field that is configured in the file **CTI_PT_DEF.CFG** for that response point.<br><br>The length of the field will be the maximum length that is defined in the file PRT_API.H in the structure PRT_ITEM_STRUCT for that field. |
| Status Value | The ASCII Text string representing a successful or failed transaction that is configured in the file **CTI_PT_DEF.CFG** will immediately follow the PRT Record field in the point. |

The CIMPLICITY ANALOG array points used to send a response must contain the following fields when populated:

| | |
|---|---|
| PRT Field Value | The first element of the array point will contain the translated value of the PRT record field that is defined in the file **CTI_PT_DEF.CFG**. |
| Status Value | The ASCII Text string representing a successful or failed transaction will immediately follow the PRT Record field in the point. |

The Success/Failure strings are defined in the file **CTI_PT_DEF.CFG**.

# Application Startup

## System Startup

Whenever the system is started/rebooted, logical names should be defined at the system level for use by the CTI software. (For CIMPLICITY HMI for Windows NT, logicals are defined in the **`%SITE_ROOT%\data\log_names.cfg`** file.)

Datagram Queue Size Logical (CTI_DGQ_SIZE) may be defined to set the size of the datagram queue. If this logical is not defined, the default datagram queue size of 10 will be used.

This logical may be defined in the **`%SITE_ROOT%\data\log_names.cfg`** file.

# CIMPLICITY for NT Startup

In order for CTI to start automatically with the other CIMPLICITY processes, it must be configured in the LOGPROC, NODE_LOGPROC, PHYSPROC, SERVICE, and MASTER.MCP files as specified in the Site Configuration section of this document.

# Point Updates

When CTI is started, it will ignore the first point value that is received in response to its on-change shopping list for points that are configured to ignore first response in **CTI_PT_DEF.CFG**. When an application requests on-change updates from Point Management, Point Management will respond with the current value of the point and then send new values whenever the value changes. Because the current value of the point may be old, it is ignored.

If a point configured to ignore first response becomes unavailable while CTI is running, typically because a device became unavailable, CTI will ignore the first value it receives when the point becomes available again because this value may be old as well.

# Transitions Between Regions

Points may be configured to move items between PRT regions or insert items into PRT regions. A point configured with a Destination Region in the **CTI_PT_DEF.CFG** file will move an existing item to the Destination Region after the item is updated if moves are allowed. CTI will only move the item to the Destination Region if is not already in the Destination region and if no Prior Region is configured or if the item is at the head of the configured Prior Region at the time the item is modified. If CTI is unable to move the item into the Destination Region or the item is not in the Prior Region, the configured Transition Failure Point will be set to '1'. It is up to the user to reset this point value.

If a point is configured to allow new items to be added, CTI will add a new item to the configured Destination Region. CTI will only allow items of the type configured in **CTI_PT_DEF.CFG** for the point to be added. If the Item Type field is left blank in **CTI_PT_DEF.CFG**, any item type may be added. If CTI is unable to add the item, the Transition Failure Point will be set to '0', otherwise it will be set to '1'.

# Stale Records

When CTI receives point updates for an item, it first tries to locate the item in the Production Tracking database.

CTI input points can be keyed by either the Item ID or Reference ID. If keyed by one of these fields, the other field can be changed by the point value. For example, if a point value is keyed by the Reference ID, the Item ID may be another value in the point. If the alternate key field value in the point is different than the current value for this field, CTI tries to locate an existing item using the alternate key field value. If found, this stale record is deleted before the keyed record is updated. A message is logged in **COR_STATUS.LOG** indicating that this has happened.

For example, assume that a point value is keyed by Reference ID but also updates the Item ID. An item currently exists in Production Tracking with Reference ID 9999 and Item ID 10000. Another item currently exists in Production Tracking with Reference ID 9100 and Item ID 19000. A point value is received by CTI with a Reference ID of 9999 and an Item ID of 19000.

When CTI receives the point update, it first tries to locate an item with a reference ID of 9999. It will find the record with the Item ID 10000. Since 10000 is different than the Item ID received from the point (19000), CTI will try to find an item in Production

Tracking with an item ID of 19000. It will find the item with Reference ID 9100. This record is no longer correct, so it is deleted from Production Tracking. The item record for reference ID 9999 is then updated with an Item ID of 19000.

CTI will process point updates keyed by Item ID in the same manner; it will try to locate stale records for the Reference ID received in the point.

# Run-Time Diagnostics

The CTI software provides run-time diagnostics to assist in determining application, configuration, or software problems.

## Debugging Logicals

The CTI subsystem was compiled with debugging statements to assist debugging at run time. Each CTI process can turn on or turn off (default) its debugging statements by defining a logical name **<PROCESS_ID>_DEBUG. PROCESS_ID** is the name of the process as configured in the SERVICE file. There are two levels (low or high) of debugging output, and the debugging level is the value (1 for low or 2 for high) of the logical name **<PROCESS_ID>_DEBUG**. The low (less verbose) level will print selected applications messages (intended to be used for application and configuration debugging). The high level (detailed and verbose) will print all the available debugging messages (intended to be used by developers to debug the programming logic),.

The debugging messages will logged in the file **<PROCESS_ID>.DBG**, and the logged file will reside in the directory specified by the logical name LOG_PATH. The logical must be defined BEFORE the process starts; the debug file may be viewed AFTER the process ends.

Example: There is a CTI process configured as CTI_RP_1. If you desire to turn on the high level debugging for the process, do the following at the DCL command line before starting the process CTI_RP_1 :

Add the following line to **%SITE_ROOT%\data\log_names.cfg**:

```
CTI_RP_1_DEBUG|P|default|10|2
```

## COR_STATUS Messages

The CTI processes may detect some exceptional errors or warning errors, and log these errors to the project's standard CIMPLICITY status log file, found in **%LOG_PATH%\COR_RECSTAT.CLG**. These errors may be detected by the CTI custom software or from API software.

To view the status log file, use the *Status Log* in the CIMPLICITY menu.

The CTI-messages are reported in **COR_STATUS.LOG** with a source of CTI_SOURCE. The messages and suggested corrective actions are described in this section.

| Message | Suggested Corrective Action |
|---|---|
| Cannot open <PATH:filename> file : | Configuration error. The filename does not exist, or it is not in the directory specified by the PATH. |
| Record number <REC_NUM> of <FILE_NAME> has too few fields | Configuration error. The specified record contains an invalid number of fields for the specified configuration file. Refer to the Configuration File section of this manual for a description of the fields required for each Configuration file. |
| Can not find any field layout for this process | Configuration error. There must be at least 1 field layout for this CTI process. |
| Can not find any field layout for point id <POINT_ID> | Configuration error. There must be at least 1 field layout for each <POINT_ID>. |
| Can not find the keyword for point id <POINT_ID> in field layout | Configuration error. Make sure that there is a keyword specified in the field layout for this <POINT_ID>. |
| "point id <POINT_ID> is not configured in base CIMPLICITY" | Configuration error. Add this <POINT_ID> into the base CIMPLICITY system. |
| point id <POINT_ID> is not a valid point type | Configuration error. The <POINT_ID> must be configured as either a TEXT POINT or ANALOG POINT in the base CIMPLICITY. |
| unexpected datagram received | Unexpected run-time error. Save the COR_STATUS.LOG file and bring it to the attention of GE-FANUC. |
| "unexpected point <POINT_ID> received" | Unexpected run-time error. Save the COR_STATUS.LOG file and bring it to the attention of GE-FANUC. |
| Point <POINT_ID> is unavailable | This is a warning message. The PLC was down, or the communication between the Device Communication process and the PLC was broken. |
| Fails to allocate dynamic memory | This is an error message. This process was unable to allocate memory. |
| Required field is missing on record %d of %s | A field is missing from a record of the configuration files. |
| Fail to set response point %s: | Unable to write to the specified point. |
| CTI Q-sort Stack overflow | The Q-sort stack is full and an attempt was made to push another item onto it. The constant QSORT_SS_SIZE in the header file CTI_QSORT.H must be increased and the CTI module must be rebuilt. |
| Point <POINT_ID> not found in the Point Definition table | The specified point ID is referenced in a CTI configuration file but is not configured in the CTI_PT_DEF.CFG file. |
| Fatal problem using PRT_API | The PRT API returned an error when CTI attempted to use it. |
| Item: <ITEM_KEY> was not found in the PRT database | The ITEM_KEY used to locate a PRT record did not correspond to any record in the PRT database. Verify that the ITEM_KEY is specified properly in the CTI Configuration files. |

| | |
|---|---|
| Failed to translate logical <LOGICAL_NAME> | The CTI received an error while attempting to obtain the logical <LOGICAL_NAME>. Verify that the specified logical is configured properly. |
| NULL point ID received from Point Management | Unexpected run-time error. Save the COR_STATUS.LOG file and bring it to the attention of GE-FANUC. |
| Point <POINT_ID> has too few elements | The specified POINT_ID is has too few elements for the field layout that is configured. The field layout for response points is defined in the file CTI_PT_DEF.CFG. The field layout for received points is defined in the file CTI_LAYOUT.CFG. |
| Response field <FIELD_NAME> not defined in layout file | Fields that are specified as response point fields must be one of the fields that is defined to be received by CTI for that point. |

# Appendix G - PRT Logging Configuration

## About PRT Data Logging

PRT data logging uses the ODBC Data source configured for CIMPLICITY alarm logging. Configuration consists of editing your project's **`datalog.idt`** and **`data_field.idt`** files (see "Configuration Requirements" for directions on this process) to create the tables and table fields necessary for the PRT attributes you wish to log.

## PRT Logging Format

The following topics cover the PRT logging format.

### Tables

The following three tables will be used to hold any logged PRT data. They must be configured in the **`datalog.idt`** file in order for any of their fields to be logged. In application, each table name will have PRT replaced with the base name of the particular PRT Service whose attributes are being logged (for example, PRT_GRP for a PRT system whose service name is APRT_DC will be APRT_GRP). The tables are independent of one another, so only those necessary for the specific data you want logged need to be configured.

#### *PRT_GRP*

The PRT_GRP table will contain data logged concerning any PRT Group.

#### *PRT_ITEM*

The PRT_ITEM table will contain data logged concerning any PRT Item or Item attribute.

#### *PRT_REG*

The PRT_REG table will contain data logged concerning any PRT Region.

## Table Fields

The following list details the allowed fields, which can be logged by PRT to each of the three tables. These fields are configured by editing the **`data_field.idt`** file. The logging configuration requires no specific fields, so you may use only those fields your application requires.

## PRT_GRP

### *Event_Type*

Text description of the type of event which triggered logging.

### *Group_Id*

Name of the group for which the event occurred.

### *Ext_Hold_Active*

External hold state of the group.

### *Ext_Hold_Reason*

Text description of reason for external hold activation.

### *Ext_Service_Id*

Name of the external process, which modified the group's status mask.

### *Comment*

Extra data sent to PRT to describe the changes made.

### *User_Id*

Identifies the agent, which modified the group's status mask.

## PRT_ITEM

### *Event_Type*

Text description of the type of event which triggered logging.

### *Entry_Time*

Time the Item entered the current region.

### *Modify_Bitmask*

Mask whose bits identify which tracking fields for this Item were modified.

### *Region_1_Id*

Name of exited region, in the case of a move, otherwise name of region, which contains modified Item.

### *Region_1_Loc*

Location of Item in exited region, in the case of a move, otherwise location in region of modified Item.

### *Region_2_Id*

Name of entered region, in the case of a move.

### *Region_2_Loc*

Location of Item in entered region, in the case of a move.

### *Item_Tracking_Type*

Types: Serialized, Non-serialized

### *Item_Id*

Item modified.

### *Reference_Id*

RefId of the modified Item.

### *Item_Type_Id*

Item type of the modified Item.

### *Assoc_Item_Id*

Id of serialized item with which this Item is associated.

### *Item_Status*

Items status mask.

### *Int_Hold_Active*

Internal Hold State.

### *Int_Hold_Reason*

Text Description of reason for internal hold activation.

### *Ext_Hold_Active*

External hold state.

### *Group_Id*

Name of group associated with activating external hold.

### *Num_Valid_Atts*

Number of valid Item attributes.

### *Att_<x>_Name*

<x> = 1 to Num_Valid_Atts. Names of each valid attribute.

### *Att_<x>_Value*

<x> = 1 to Num_Valid_Atts. Values of each valid attribute.

### *Comment*

Extra data sent to PRT to describe the changes made.

### *User_Id*

Identifies the agent which modified the Item.

## PRT_REG

### *Event_Type*

Text description of the type of event which triggered logging.

### *Region_Id*

Name of modified region.

### *Region_Status*

Region status mask.

### *Comment*

Extra data sent to PRT to describe the changes made.

### *User_Id*

Identifies the agent which modified the Region.

# Configuration Files

The nature of PRT database logging logging is such that the fields for a given entry in the idt files will be the same across most platforms and projects. Therefore, the following sections simply contain sample files showing the configurations for the maximum set of loggable data. These are not replacements for your existing files, but rather should be merged with them. In your own configuration, you need only copy out the lines from these files necessary for the specific PRT data you wish to log and add them to your own existing datalog.idt and data_field.idt files. Remember to replace "PRT" with the base name of your data server.

## Datalog.idt

The following is a sample datalog.idt file.

```
|-* IDT file generated by IDTPOP utility v1.0
* RECORD: DATALOG PARAMETERS FOR DATA LOG FILES
*
* SAMPLE FILE - merge with datalog.idt
*
* 0 FILE_ID             log file name
* 1 SERVICE_ID          Owner service id for the file
* 2 max_rec_size        max size of a record in bytes
* 3 max_recs            max number of records before rollover
* 4 alarm_rec           threshold for alarm generation
* 5 rollover_recs       number of records to keep on rollover
* 6 auto_rollover       TRUE=rollover on number of records logged
* 7 daily_rollover      TRUE=rollover on time_of_day (sync time)
* 8 add_stamp           TRUE=add time stamp to each log entry
* 9 rollover_cmd        DOS cmd to execute on rollover
* 10 timed_rollup        TRUE=rollover on timed basis (periodic)
* 11 archive_id          two character archive id
* 12 compress_arc        TRUE=compress archive file
* 13 batch_pt_id         batch point id associated with log file
* 14 dbms_id             Database in DBMS_DEF file
* 15 table_type          one of A/D/G/X indicating log type
* 16 space_rollover      TRUE if rollover on disk full
* 17 gated_rollover      TRUE if rollover on digital point
* 18 redund_rollover     TRUE if rollover on redundant transition
* 19 rollover_dev        device to check for disk full
* 20 rollover_space      limit (in Kb) for disk full rollover
* 21 gate_point          point to gate rollover
* 22 gate_condition      1=HIGH, 2=LOW, 3=TRANSITION
* 23 rollover_units      5=DAYS, 4=HOURS, 3=MINUTES
* 24 rollover_period     frequency for periodic rollover
* 25 rollover_time       time for synch rollover in min from 00:0
* 26 rollover_export     0=none, 1=all, 2=count, 3=time
* 27 rollover_purge      0=none, 1=all, 2=count, 3=time, 5=exp+al
* 28 rollover_exec       0=none, 1=SQL, 2=DOS, 3=both
* 29 export_file         export target file
* 30 purge_file          purge export target file
* 31 export_format       'C'=CSV
* 32 purge_format        'C'=CSV
* 33 export_range        time/# of records to export
* 34 purge_range         time/# of records not to purge
* 35 sql_command         SQL cmd to execute on rollover
```

```
*
PRT_GRP|DL_RP|200|500|0|100|1|0|1||0|PG|0||$LOGGING|X|0|0|0||-
0||0|5|1|0|0|3|0|||C|C|0|86400|
PRT_ITEM|DL_RP|960|1000|0|100|1|0|1||0|PI|0||$LOGGING|X|0|0|-
0||0||0|5|1|0|0|3|0|||C|C|0|86400|
PRT_REG|DL_RP|130|500|0|100|1|0|1||0|PR|0||$LOGGING|X|0|0|0||-
0||0|5|1|0|0|3|0|||C|C|0|86400|
```

## Data_field.idt

The following is a sample **`data_field.idt`** file:

```
|-* IDT file generated by IDTPOP utility v1.0
* RECORD: DATA_FIELD DLRP Field Definitions
*
* SAMPLE FILE - merge with data_field.idt
*
*  0 TABLE_ID            DLRP Table Identifier
*  1 field_id            DLRP Field to be logged
*  2 cimp_type           CIMPLICITY Data Type of Field
*  3 field_len           Size of Field, in cimp_type units
*  4 offset              Offset of field in DLAP buffer
*
PRT_GRP|event_type|12|14|1
PRT_GRP|group_id|12|16|15
PRT_GRP|ext_hold_active|12|1|31
PRT_GRP|ext_hold_reason|12|40|32
PRT_GRP|ext_service_id|12|32|72
PRT_GRP|comment|12|40|104
PRT_GRP|user_id|12|32|144
PRT_ITEM|event_type|12|14|1
PRT_ITEM|entry_time|12|21|15
PRT_ITEM|modify_bitmask|12|4|36
PRT_ITEM|region_1_id|12|16|40
PRT_ITEM|region_1_loc|12|5|56
PRT_ITEM|region_2_id|12|16|61
PRT_ITEM|region_2_loc|12|5|77
PRT_ITEM|tracking_type|12|3|82
PRT_ITEM|item_id|12|20|85
PRT_ITEM|reference_id|12|20|105
PRT_ITEM|item_type_id|12|16|125
PRT_ITEM|assoc_item_id|12|20|141
PRT_ITEM|item_status|12|8|161
PRT_ITEM|int_hold_active|12|1|169
PRT_ITEM|int_hold_reason|12|40|170
PRT_ITEM|ext_hold_active|12|1|210
PRT_ITEM|group_id|12|16|211
PRT_ITEM|num_valid_atts|12|3|227
PRT_ITEM|att_1_name|12|16|230
PRT_ITEM|att_1_value|12|16|246
PRT_ITEM|att_2_name|12|16|262
PRT_ITEM|att_2_value|12|16|278
PRT_ITEM|att_3_name|12|16|294
PRT_ITEM|att_3_value|12|16|310
PRT_ITEM|att_4_name|12|16|326
PRT_ITEM|att_4_value|12|16|342
```

```
PRT_ITEM|att_5_name|12|16|358
PRT_ITEM|att_5_value|12|16|374
PRT_ITEM|att_6_name|12|16|390
PRT_ITEM|att_6_value|12|16|406
PRT_ITEM|att_7_name|12|16|422
PRT_ITEM|att_7_value|12|16|438
PRT_ITEM|att_8_name|12|16|454
PRT_ITEM|att_8_value|12|16|470
PRT_ITEM|att_9_name|12|16|486
PRT_ITEM|att_9_value|12|16|502
PRT_ITEM|att_10_name|12|16|518
PRT_ITEM|att_10_value|12|16|534
PRT_ITEM|att_11_name|12|16|550
PRT_ITEM|att_11_value|12|16|566
PRT_ITEM|att_12_name|12|16|582
PRT_ITEM|att_12_value|12|16|598
PRT_ITEM|att_13_name|12|16|614
PRT_ITEM|att_13_value|12|16|630
PRT_ITEM|att_14_name|12|16|646
PRT_ITEM|att_14_value|12|16|662
PRT_ITEM|att_15_name|12|16|678
PRT_ITEM|att_15_value|12|16|694
PRT_ITEM|att_16_name|12|16|710
PRT_ITEM|att_16_value|12|16|726
PRT_ITEM|att_17_name|12|16|742
PRT_ITEM|att_17_value|12|16|758
PRT_ITEM|att_18_name|12|16|774
PRT_ITEM|att_18_value|12|16|790
PRT_ITEM|att_19_name|12|16|806
PRT_ITEM|att_19_value|12|16|822
PRT_ITEM|att_20_name|12|16|838
PRT_ITEM|att_20_value|12|16|854
PRT_ITEM|att_21_name|12|16|870
PRT_ITEM|att_21_value|12|16|886
PRT_ITEM|att_22_name|12|16|902
PRT_ITEM|att_22_value|12|16|918
PRT_ITEM|att_23_name|12|16|934
PRT_ITEM|att_23_value|12|16|950
PRT_ITEM|att_24_name|12|16|966
PRT_ITEM|att_24_value|12|16|982
PRT_ITEM|att_25_name|12|16|998
PRT_ITEM|att_25_value|12|16|1014
PRT_ITEM|att_26_name|12|16|1030
PRT_ITEM|att_26_value|12|16|1046
PRT_ITEM|att_27_name|12|16|1062
PRT_ITEM|att_27_value|12|16|1078
PRT_ITEM|att_28_name|12|16|1094
PRT_ITEM|att_28_value|12|16|1110
PRT_ITEM|att_29_name|12|16|1126
PRT_ITEM|att_29_value|12|16|1142
PRT_ITEM|att_30_name|12|16|1158
PRT_ITEM|att_30_value|12|16|1174
PRT_ITEM|att_31_name|12|16|1190
PRT_ITEM|att_31_value|12|16|1206
PRT_ITEM|att_32_name|12|16|1222
PRT_ITEM|att_32_value|12|16|1238
```

```
PRT_ITEM|att_33_name|12|16|1254
PRT_ITEM|att_33_value|12|16|1270
PRT_ITEM|att_34_name|12|16|1286
PRT_ITEM|att_34_value|12|16|1302
PRT_ITEM|att_35_name|12|16|1318
PRT_ITEM|att_35_value|12|16|1334
PRT_ITEM|att_36_name|12|16|1350
PRT_ITEM|att_36_value|12|16|1366
PRT_ITEM|att_37_name|12|16|1382
PRT_ITEM|att_37_value|12|16|1398
PRT_ITEM|att_38_name|12|16|1414
PRT_ITEM|att_38_value|12|16|1430
PRT_ITEM|att_39_name|12|16|1446
PRT_ITEM|att_39_value|12|16|1462
PRT_ITEM|att_40_name|12|16|1478
PRT_ITEM|att_40_value|12|16|1494
PRT_ITEM|comment|12|40|1510
PRT_ITEM|user_id|12|32|1550
PRT_REG|event_type|12|14|1582
PRT_REG|region_id|12|16|1596
PRT_REG|region_status|12|8|1612
PRT_REG|comment|12|40|1620
PRT_REG|user_id|12|32|1660
```

# Appendix H – Configuring PRT Client Startup

## About PRT Client Startup Configuration

In order to have the PRT Client application start when CIMPLICITY starts the following configuration files must be modified. Each file needs to be changed in the Master directory.

An example of each file follows showing what needs to be added to that file.

The files are:

- `Physproc`
- `Logproc`
- `Node_Logproc`
- `Service`
- `Master.mcp`

## PRT Client Configuration Files to be Modified

PRT Client configuration files to be modified include:

- Physproc
- Logproc
- Node_Logproc
- Service
- Master.mcp

## Physproc

```
|-* IDT file generated by IDTPOP utility v1.0
* RECORD: PHYSPROC PROCESS NAMES ON SPECIFIC NODES
*
* 0 NODE_ID          Decnet Node Name
* 1 PROCESS_ID       Logical Process Name
* 2 object_name      Decnet Object Name
* 3 order            Redundant or Allocated Process Order
*
MASTER|PRT_CLNT|PRT_CLNT|0
```

## Logproc

```
|-* IDT file generated by IDTPOP utility v1.0
* RECORD: LOGPROC LOGICAL PROCESS IDENTIFIERS
*
* 0 PROCESS_ID Process Identifier
* 1 process_type_id    Process Type Identifier
* 2 pm_flags           Bits(last 5):(AP)(RSTAL)(LD)(NOAL)(WT)
* 3 description        Process Description
*
PRT_CLNT|RESIDENT|1|PRT Client
```

## Node_Logproc

```
|-* IDT file generated by IDTPOP utility v1.0
* RECORD: NODE_LOGPROC PROCESSES ON SPECIFIC NODES
*
* 0 NODE_ID          Decnet Node Name
* 1 PROCESS_ID       Process Identifier
* 2 image_name       Executable Image name
* 3 priority         Process Priority
* 4 base_quan        Base quantity
* 5 max_quan         Max Quantity
* 6 astlm            AST Limit
* 7 biolm            BIO Limit
* 8 bytlm            BYTE Limit
* 9 cpulm            CPU Limit
* 10 diolm           DIO Limit
* 11 enqlm           ENQ Limit
* 12 fillm           Fil Limit
* 13 pgflquota       Page file quota
* 14 prcml           PRCLM (field name is a typo)
* 15 tqelm           TQELM
* 16 wsdefault       Working set extet
* 18 wsquota         Working set quota
* 19 sys_priv_mask   System wide process privilege mask
*
MASTER|PRT_CLNT|BSM_ROOT:[exe]PRTClient.exe|20|1|1|80|50|50000|0|50|2000|100|50000|-
10|50|99|999|9999|-1
```

## Service

```
|-* IDT file generated by IDTPOP utility v1.0
* RECORD: SERVICE SERVICE IDENTIFIERS
*
* 0 SERVICE_ID      Service Identifier
* 1 SUBSYS_ID       Subsystem Identifier
* 2 PROCESS_ID      Process Identifier
*
PRT_CLNT|PRT|PRT_CLNT
```

## Master.mcp

```
PRT_CLNT
```

# Transition Type Overview

The following gives an overview how each of the Transition Type Codes function.

Transition types include:

1. Limit Switch
2. Type Detector
3. Item Identifier
4. Reference Identifier.
5. Associate Reference ID to Item
6. Associate Item Type to Item.
7. Positive Edge Trigger
8. Negative Edge Trigger

Transition Type 1.

## Limit Switch

The first location of the source region is checked for items.

Actions triggered by what items exist are:

If: No items exist in the source region

Then: No items are moved.

If: Multiple items exist at a location

Then: All items at the location move.

Actions triggered by the specified handling of a moved item are:

If: Not handled along the route

Then: The item moves and

A message is logged warning the user.

If: Not allowed in the destination region

Then: The item is deleted and

A message is logged warning the user.

**Guidelines for `Limit Switch` include:**

1. The transition point may be an integer or character string point.
2. The first 36 characters of a setpoint, if configured, are:

| | |
|---|---|
| First 16 characters | Item type |
| Next 20 characters | Either the:<br>▪ Item ID–for a serialized item<br>▪ Associated or parent item ID–for a non-serialized item. |

Transition Type 2.

## Type Detector

Tracker cross-references the given integer with the item type.

**Important:** The **PRT_TYPE TRN** file must be configured to describe the cross-reference between the integer entered and the item type to be moved.

Actions triggered in the region are:

If: The item type is found in the source region

Then: The item is moved to the destination region.

If: The source region is a disperse region

Then: Only the item whose item type is cross-referenced with the given integer moves.

If: Items exist at a location and the source region is not a disperse region

Then: All items are moved.

Actions triggered by the handling of a moved item are:

If: Not handled along the route

Then: The item moves and

A message is logged warning the user.

If: Not allowed in the destination region

Then: The item is deleted and

A message is logged warning the user.

Transition Type 3.

## Item Identifier

Tracker searches for the item with the given item ID in the entire model.

Actions triggered by Tracker finds in the entire model are:

If: The item is not found.

Then: A message is logged.

If: The source region is a disperse region.

Then: Only the given item is moved.

If: Multiple items exist at a location and the source region is not a disperse region.

Then: All items at the location are moved.

Actions triggered by the handling of a moved item are:

If: Not handled along the route

Then: The item moves and

A message is logged warning the user.

If: Not allowed in the destination region

Then: The item is deleted and

A message is logged warning the user.

**Note:** If the point value begins with a $ the remainder of the point value is used to identify the item type to move.

The transition type code becomes a Type Detector at this point.

Tracker searches for the given item type and moves the item to the destination.

Again, in this case:

If: The source region is a disperse region.

Then: Only the item with the given item type is moved.

If: Multiple items exist at a location and the source region is not a disperse region.

Then: All items at the location are moved.

Actions triggered by the handling of a moved item are:

If: Not handled along the route

Then: The item moves and

A message is logged warning the user.

If: Not allowed in the destination region

Then: The item is deleted and

A message is logged warning the user.

Transition Type 4.

## Reference Identifier

Tracker searches for the item with a given reference ID in the entire model.

Actions triggered by what Tracker finds in the entire model are:

If: The item is not found.

Then:. A message is logged.

If: The source region is a disperse region.

Then: Only the given item is moved.

If: Multiple items exist at a location and the source region is not a disperse region.

Then: All items at the location are moved.

Actions triggered by the handling of a moved item are:

If: Not handled along the route

Then: The item moves and

A message is logged warning the user.

If: Not allowed in the destination region

Then: The item is deleted and

A message is logged warning the user.

**Note:** If the point value begins with a $ the remainder of the point value is considered the item type to move.

The transition type code becomes a Type Detector at this point.

Tracker searches for the item with the given item type and moves it to the destination.

Again, in this case:

If: The source region is a disperse region.

Then: Only the given item is moved.

If: Multiple items exist at a location and the source region is not a disperse region.

Then: All items at the location are moved.

Actions triggered by the specified handling of a moved item are:

If: Not handled along the route

Then: The item moves and

A message is logged warning the user.

If: Not allowed in the destination region

Then: The item is deleted and

A message is logged warning the user.

Transition Type 5.

## Associate Reference ID to Item

Tracker checks the first location of the source region for items.

Tracker moves all serialized items in the location to the destination region gives each serialized item a reference ID of the point.

**Important:** Reference ID's are unique. Therefore, if multiple serialized items are moved the last serialized item will exist at the destination region with all previous items in that region being overwritten.

Non serialized items do not have reference ID's so they are moved and not affected by the reference ID.

Actions triggered by what serialized items exist are:

If: No serialized items are in the source region.

Then: A message is logged warning the user.

If: Multiple serialized items exist at a location.

Then: All items are moved.

Actions triggered by the handling of a moved item are:

If: Not handled along the route

Then: The item moves and

A message is logged warning the user.

If: Not allowed in the destination region

Then: The item is deleted and

A message is logged warning the user.

Transition Type 6.

## Associate Item Type to Item

Tracker searches the source region for the item type **$UNKNOWN**.

If **$UNKNOWN** is found the item type is changed to what the user entered.

Actions triggered by the result of the search for **$UNKNOWN** are:

If: No items with the **$UNKNOWN** item type are in the source region.

Then: A message is logged warning the user.

If: The source region is a disperse region.

Then: Only the item with the **$UNKNOWN** item type will move.

If: Multiple items exist at a location and the source region is not a disperse region..

Actions triggered by the handling of a moved item are:

If: Not handled along the route

Then: The item moves and

A message is logged warning the user.

If: Not allowed in the destination region

Then: The item is deleted and

A message is logged warning the user.

Transition Type 7.

## Positive Edge Trigger

Tracker checks the first location of the source region for items.

Actions triggered by what items exist are:

If: No items are in the source region.

Then: A message is logged warning the user.

If: Multiple items exist at a location.

Then: All items are moved.

Actions triggered by the specified handling of a moved item are:

If: Not handled along the route

Then: The item moves and

A message is logged warning the user.

If: Not allowed in the destination region

Then: The item is deleted and

A message is logged warning the user.

**Guidelines for `Positive Edge Trigger` include:**

1. The transition point may be an integer or character string point.
2. The item transitions when the point goes from low (integer – 0, character string – null string) to high ( integer – non-zero, character string – non null string)
3. The first 36 characters of a setpoint, if configured, are:

| | |
|---|---|
| First 16 characters | Item type |
| Next 20 characters | Either the:<br>▪ Item ID–for a serialized item<br>▪ Associated or parent item ID–for a non-serialized item. |

Transition Type 8.

## Negative Edge Trigger

Tracker checks the first location of the source region for items.

Actions triggered by what items exist are:

If: No items exist in the source region.

Then: No items are moved.

If: Multiple items exist at a location.

Then: All items are moved.

Actions triggered by the specified handling of a moved item are:

If: Not handled along the route

Then: The item moves and

A message is logged warning the user.

If: Not allowed in the destination region

Then: The item is deleted and

A message is logged warning the user.

**Guidelines for `Negative Edge Trigger` include:**

1. The transition point may be an integer or character string point.
2. The item transitions when the point goes from high( integer – non-zero, character string – non null string) to low (integer – 0, character string – null string)
3. The first 36 characters of a setpoint, if configured, are:

| | |
|---|---|
| First 16 characters | Item type |
| Next 20 characters | Either the:<br>▪ Item ID–for a serialized item<br>▪ Associated or parent item ID–for a non-serialized item. |

# Appendix I – Attribute Counts by Region

## About Attribute Counts by Region

Production Tracking counts PRT items based on:

- Region(s)
- Item class
- Various item attributes criteria

An item is included in a count if its attribute matches a configured set of criteria. In order for an item to be included in the count, the attribute value must be compared against a user-defined value. The comparison can use any of the following user-defined operators:

| | |
|---|---|
| == | Equals |
| != | Not Equals |
| < | Less Than |
| <= | Less Than Or Equal |
| > | Greater Than |
| >= | Greater Than Or Equal |
| @ | Between |
| !@ | Not Between |
| % | Exists |
| $ | Contains |

The comparison may be:

- Numeric(N)
- Alphanumeric(A)
- Float(F)

Alphanumeric comparison values can specify the wildcard character(“?”).

Multiple region names can be specified by separating them with commas.

**Example**

Specify "Source Regions 1 and 2 and Distribution Regions 1 and 2":

SRC1, SRC2,DST1,DST2.

When using @ (Between) or !@ (Not Between), separate the high and low values with a comma.

**Example**

Specify "Between 2 and 3":

|@|2,3|

Counts are stored in CIMPLICITY HMI analog non-array points. These points are initialized at startup and are updated as items enter and exit a region. When the process first starts it:

- Performs a count of all items in the configured regions
- Initialize the points with the count values.

Utilizes a configured reset point; when the reset point updates, PRTCNT will automatically recount all regions. When an item is modified in a configured region, PRTCNT will recount all counters associated with that region.

# Configuration Files

The program looks for the configuration file *(PRTCNT.CF*G) in

SC_PATH\data.

### Attribute Counts by Region configuration file

Record 1 lists the reset point.

Data on each succeeding line displays as follows:

**`RegionName|ItemClass|AttribName|StartChar|Length|Operator|Value|CompareType|PointID`**

| | |
|---|---|
| Region | Name of the region(s) where the items reside; separate multiple regions by commas. |
| Item Class | Item class of the items to be included. |
| Attrib Name | Name of the attribute whose value is to be compared. |
| Start Char | Starting character within the attribute value. (Allows comparison of substrings) |
| Length | Number of characters to include in the comparison. |
| Operator | The type of the comparison to be performed between the attribute and the user specified value<br>(==,!=,>,>=,<,<=,%(exists),$(contains),@(between),!@). |
| Value | Value to be compared against attribute value. For alphanumeric comparisons, this value can contain the wildcard character “?”.<br>For @, !@ operators, separate two (2) values with a comma. |
| Compare type | Type of Comparison, N (Numeric integer),A (Alphanumeric) and F (real). Attribute will be converted to this type before comparison. |
| Point Id | Id of CIMPLICITY analog non-array point which will hold the count. |

Following is a listing of a sample configuration file.

Sample configuration file:

```
|-*
PRTCNT_RESET
*Multiple Region Counts
SRC1,SRC2,DST1,DST2|VEHICLE|COLOR|0|10|!=|BLUE|A|BIG_CNT1
SRC1,SRC2,DST1,DST2|CARRIER|CYCLE_CNT|0|10|!=|2|F|BIG_CNT2
* Region SRC2
SRC2|CARRIER|CYCLE_CNT|0|10|==|2|F|SRC2_CNT1
SRC2|CARRIER|CYCLE_CNT|0|10|!=|2|F|SRC2_CNT2
SRC2|CARRIER|CYCLE_CNT|0|10|>|2|F|SRC2_CNT3
SRC2|CARRIER|CYCLE_CNT|0|10|>=|2|F|SRC2_CNT4
SRC2|CARRIER|CYCLE_CNT|0|10|<|2|F|SRC2_CNT5
SRC2|CARRIER|CYCLE_CNT|0|10|<=|2|F|SRC2_CNT6
SRC2|CARRIER|CYCLE_CNT|0|10|$|2|F|SRC2_CNT7
SRC2|CARRIER|CYCLE_CNT|0|10|%||F|SRC2_CNT8
SRC2|CARRIER|CYCLE_CNT|0|10|@|2,3|F|SRC2_CNT9
SRC2|CARRIER|CYCLE_CNT|0|10|!@|2,3|F|SRC2_CNT10
* Region DST1
DST1|VEHICLE|COLOR|0|10|==|RED|A|DST1_CNT1
DST1|VEHICLE|COLOR|0|10|>|RED|A|DST1_CNT3
DST1|VEHICLE|COLOR|0|10|<|RED|A|DST1_CNT5
DST1|VEHICLE|COLOR|0|10|$|R|A|DST1_CNT7
DST1|VEHICLE|COLOR|0|10|%||A|DST1_CNT8
DST1|VEHICLE|COLOR|0|10|@|BLUE,RED,|A|DST1_CNT9
DST1|VEHICLE|COLOR|0|10|!@|BLUE,RED|A|DST1_CNT10
DST1|VEHICLE|COLOR|0|10|==|??D|A|DST1_CNT11
* Region DST2
DST2|VEHICLE|PRT_ITEM_ID|2|2|==|5|N|DST2_CNT1
DST2|VEHICLE|PRT_ITEM_ID|2|2|!=|5|N|DST2_CNT2
DST2|VEHICLE|PRT_ITEM_ID|2|2|>=|5|N|DST2_CNT4
DST2|VEHICLE|PRT_ITEM_ID|2|2|<|5|N|DST2_CNT5
DST2|VEHICLE|PRT_ITEM_ID|2|2|$|5|N|DST2_CNT7
DST2|VEHICLE|PRT_ITEM_ID|2|2|%|5|N|DST2_CNT8
DST2|VEHICLE|PRT_ITEM_ID|2|2|@|5,10|N|DST2_CNT9
DST2|VEHICLE|PRT_ITEM_ID|2|2|!@|5,10|N|DST2_CNT10
DST2|VEHICLE|PRT_ITEM_ID|0|4|==|D?0?|A|DST2_CNT11
*
```

# Setting Up

The following **.idt** files need to be modified to have PRTCNT function properly. Add records as follows:

## Logproc

```
|-* IDT file generated by IDTPOP utility v1.0
* RECORD: LOGPROC LOGICAL PROCESS IDENTIFIERS
*
*  0  PROCESS_ID           Process Identifier
*  1  process_type_id      Process Type Identifier
*  2  pm_flags             Bits(last 5):(AP)(RSTAL)(LD)(NOAL)(WT)
*  3  description          Process Description
*
PRTCNT|RESIDENT|1|Prt Attribute Counts
```

## Node_Logproc

```
|-* IDT file generated by IDTPOP utility v1.0
* RECORD: NODE_LOGPROC PROCESSES ON SPECIFIC NODES
*
*  0  NODE_ID              Decnet Node Name
* 1 PROCESS_ID            Process Identifier
* 2 image_name            Executable Image name
* 3 priority              Process Priority
* 4 base_quan             Base quantity
* 5 max_quan              Max Quantity
* 6 astlm                 AST Limit
* 7 biolm                 BIO Limit
* 8 bytlm                 BYTE Limit
* 9 cpulm                 CPU Limit
* 10 diolm                DIO Limit
* 11 enqlm                ENQ Limit
* 12 fillm                Fil Limit
* 13 pgflquota            Page file quota
* 14 prcml                PRCLM (field name is a typo)
* 15 tqelm                TQELM
* 16 wsdefault            Working set Default
* 17 wsextent             Working set extet
* 18 wsquota              Working set quota
* 19 sys_priv_mask        System wide process privilege mask
*
MASTER|PRTCNT|BSM_ROOT:[exe]prtcnt.exe|4|1|1|80|50|50000|0|50|2
000|100|50000|10|-50|
99|999|9999|-1
```

## Physproc

```
|-* IDT file generated by IDTPOP utility v1.0
* RECORD: PHYSPROC PROCESS NAMES ON SPECIFIC NODES
*
*  0  NODE_ID              Decnet Node Name
*  1  PROCESS_ID           Logical Process Name
*  2  object_name Decnet   Object Name
*  3  order Redundant or   Allocated Process Order
*
MASTER|PRTCNT|PRTCNT|0
```

## Prt_Intproc

```
|-* IDT file generated by IDTPOP utility v1.0
* RECORD: PRT_INTPROC Interested Process/Data provider mapping
*
*  0  prt_svc_id           PRT Service ID providing info. to above
*  1  INTPROC_SVC_ID       Interested Process Service ID
*  2  update_mask          1=Full, 0=Brief(all but auto move)
*
APRT_DC|PRTCNT|1
```

## Service

```
|-* IDT file generated by IDTPOP utility v1.0
* RECORD: SERVICE SERVICE IDENTIFIERS
*
*  0  SERVICE_ID           Service Identifier
*  1  SUBSYS_ID            Subsystem Identifier
*  2  PROCESS_ID           Process Identifier
*
PRTCNT|PRTCNT|PRTCNT
```

## master.mcp

```
PRTCNT
```

# Appendix J – Troubleshooting

## Troubleshooting Overview

In an effort to continually provide our customers with up-to-date information on all of our CIMPLICITY products, this guide lends solutions for problems with your Tracker system. This section will evolve as new solutions arise and new tools are developed.

# PRT Interested Process Utility

The PRT Interested Process Utility (**`prt_inttest.exe`**) is used to ensure that the Tracker system is generating interested processes. The utility is configured using a combination of CIMPLICITY and PRT files. All files are stored in the master directory and can be edited using the IDTPOP Utility to convert records to ASCII-text format. The output generated by the PRT Interested Process Utility can be viewed in a Command window from the CIMPLICITY Workbench.

The following output can be generated:

Item Update

- `reg_entry_time`
- `event_code`
- `modify_bitmask`
- `region_1_id`
- `region_1_loc`
- `region_2_id`
- `region_2_loc`
- `item_tracking_type`
- `item_id`
- `reference_id`
- `item_type_id`
- `parent_item_id`
- `item_status`
- `int_hold_active`
- `item_hold_reason`
- `ext_hold_active`
- `group_id`
- `num_atts_valid`
- `item_att_list`
- `comment`
- `user_or_svc_id`
- `action`

Region Update

- event_code
- region_id
- region_status
- comment
- user_or_svc_id
- location
- action

Error Response

- status
- err_code
- err_ref
- err_msg

# Configuring the PRT Interested Process Utility

The six files needed to configure the PRT Interested Process Utility (**`prt_inttest.exe`**) can be found in the Master directory of CIMPLICITY. The files are in binary format and must be converted to ASCII text using the IDTPOP Utility. Also, files that have overlapping fields (for example, process_id) must have identical values in order to run the utility. Sample configuration files are provided for your reference in this section.

*See "Configuration Requirements" chapter for instructions on using the IDTPOP Utility.*

The required configuration files are:

- **`physproc.dat`**
- **`logproc.dat`**
- **`node_logproc.dat`**
- **`service.dat`**
- **`prt_intproc.dat`**

The following steps provide instruction on configuring prt_inttest.exe:

**Step 1** Open Tracker project in Workbench

**Step 2** Configure physproc

**Step 3** Configure logproc

**Step 4** Configure node_logproc

**Step 5** Configure service

**Step 6** Configure prt_intproc

## *Step 1. Open Tracker Project in Workbench*

**Step 1. Open Tracker project in Workbench:**

1. Click **Start** on the Windows task bar.
2. Select Programs>CIMPLICITY>HMI>Workbench.

   The Workbench window opens.
3. Click File on the Workbench menu bar.
4. Select Open.

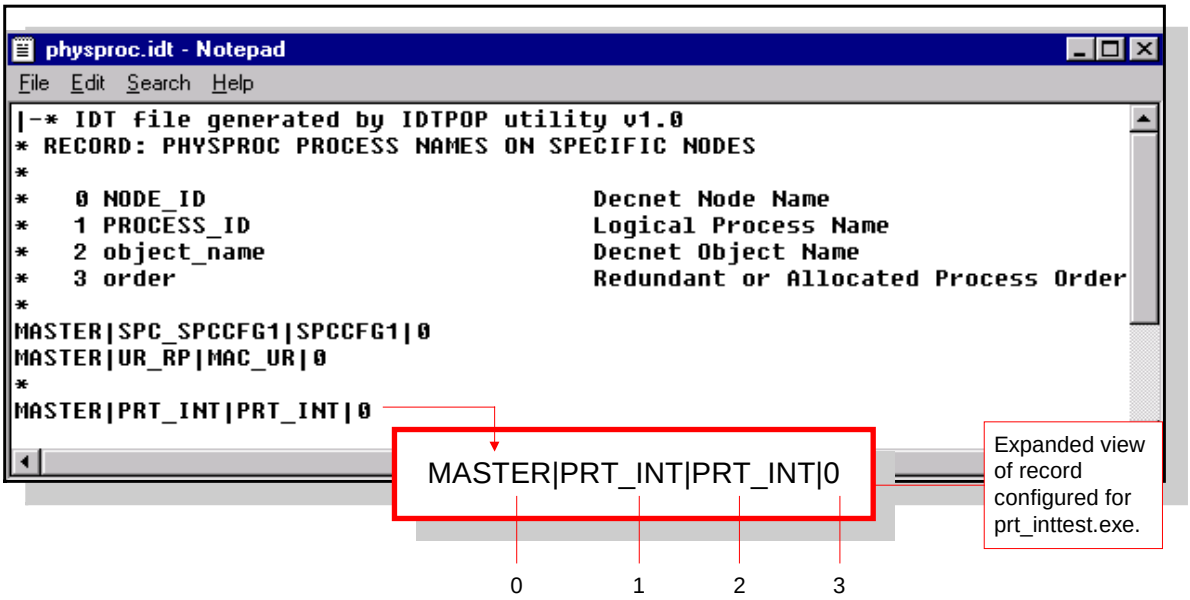
   the Open dialog box opens.
5. Navigate to and click your Tracker project.
6. Click **Open** to open your project in the Workbench window.

**Tip:** To open a project quickly, create a shortcut for the project on your Windows desktop and double-click the icon.

## *Step 2. Configure Physproc*

**Step 2. Configure physproc:**

1. Use the IDTPOP Utility to convert the file to ASCII-text using the text editor of your choice, e.g. Notepad.
2. Press **Ctrl+End** to bring you to the next available line to enter your record—see figure.



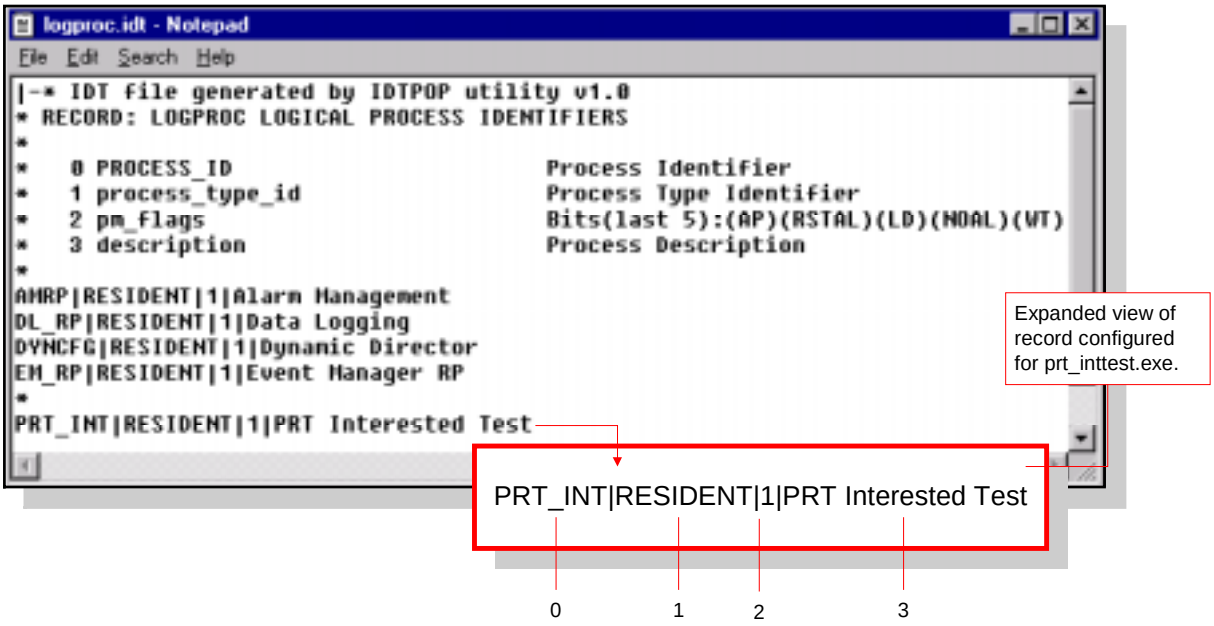

3. Type your record using the following field descriptions:

| Field No. | Field Name | Description | Entry Example |
|---|---|---|---|
| 0 | node_id | Node name, always MASTER. | MASTER |
| 1 | process_id | The name of the process that the program will look to for data. | PRT_INT |
| 2 | object_name | Object name, same as Process ID. | PRT_INT |
| 3 | order | Use default value = 0. | 0 |

4. Save the file.
5. Use the SCPOP Utility to convert file back to binary format.

### Step 3. Configure logproc:

1. Use the IDTPOP Utility to convert the file to ASCII-text using the text editor of your choice, e.g. Notepad.
2. Press **Ctrl+End** to bring you to the next available line to enter your record—see figure.




3. Type your record using the following field descriptions:

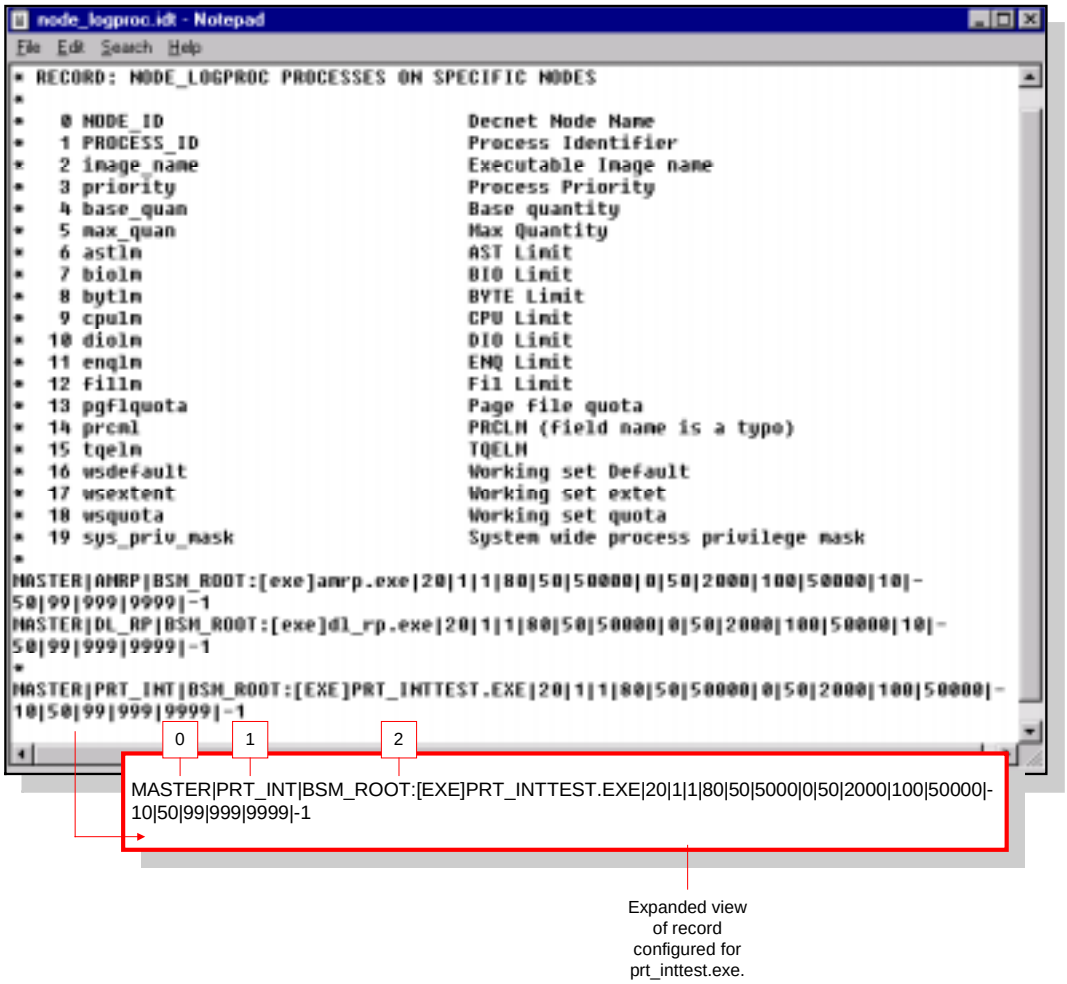
| Field No. | Field Name | Description | Entry Example |
|---|---|---|---|
| 0 | process_id | The name of the process that the program will look to for data. | PRT_INT |
| 1 | process_type_id | Process type ID; usually RESIDENT process. | RESIDENT |
| 2 | pm_flags | Use default value = 1. | 1 |
| 3 | description | Description of the process | PRT Interested Test |

4. Save the file.
5. Use the SCPOP Utility to convert the file back to binary format.

### *Step 4. Configure Node_logproc*

**To configure node_logproc:**

1. Use the IDTPOP Utility to convert the file to ASCII-text using the text editor of your choice, e.g. Notepad.
2. Press **Ctrl+End** to bring you to the next available line to enter your record—see figure.




3. For this record you are only concerned with the entries in the first three fields. Use the default values for the remaining fields, which can be found on the sample configuration.

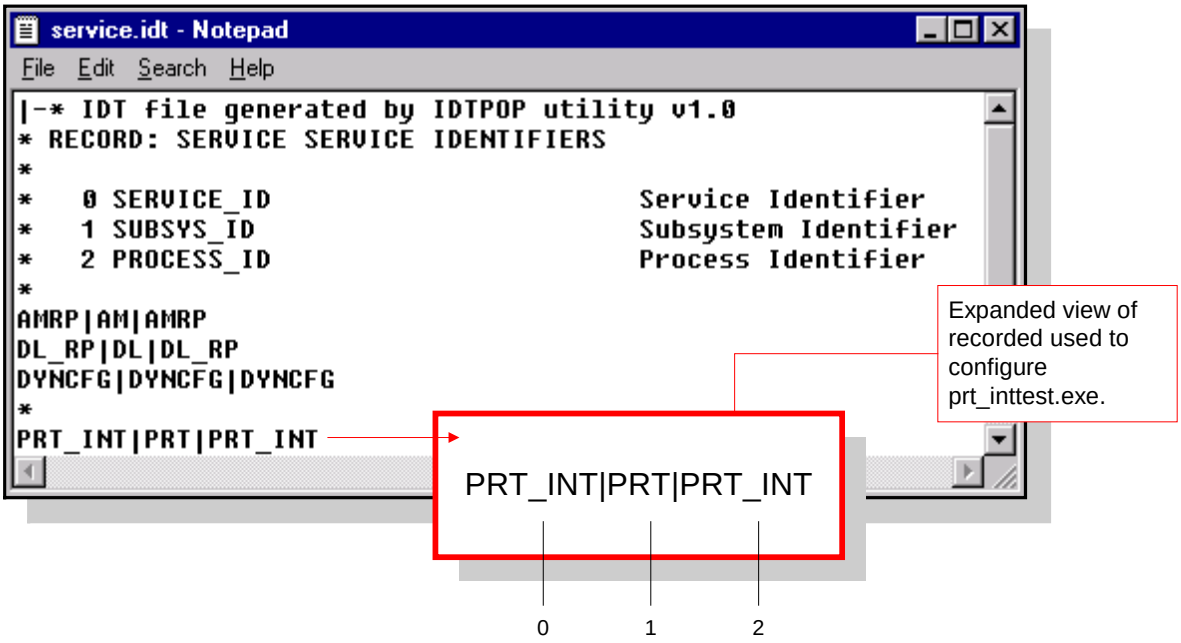
| Field No. | Field Name | Description | Entry Example |
|---|---|---|---|
| 0 | node_id | Node name, always MASTER. | MASTER |
| 1 | process_id | The name of the process that the program will look to for data. | PRT_INT |
| 2 | image_name | The name of the executable image. Configure as follows: Directory_name:[exe]prt_inttest.exe | BSM_ROOT:[EXE] PRT_INTTEST.EXE |

4. Save the file.
5. Use the SCPOP Utility to convert file back to binary format.

## *Step 5. Configure Service*

**Step 5. Configure service:**

1. Use the IDTPOP Utility to convert the file to ASCII-text using the text editor of your choice, e.g. Notepad.
2. Press **Ctrl+End** to bring you to the next available line to enter your record—see figure.




3. Type your record using the following field descriptions:

| Field No. | Field Name | Description | Entry Example |
|---|---|---|---|
| 0 | service_id | The name given to this service. | PRT_INT |
| 1 | subsys_id | The ID of the subsystem, in this case PRT. | PRT |
| 2 | process_id | The name of the process that the program will look to for data. | PRT_INT |

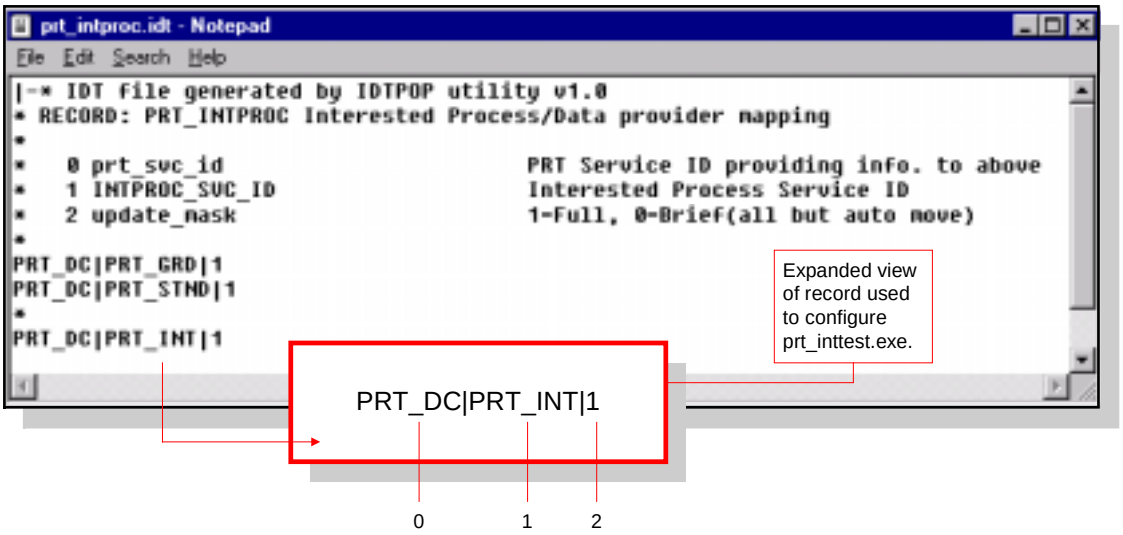
4. Save the file.
5. Use the SCPOP Utility to convert file back to binary format.

### *Step 6. Configure Prt_intproc*

| | |
|---|---|
| Field Number: | 0 |
| Description: | Unique identifier of a PRT service that will provide interested process updates. This must be identical to the ID in . |
| Data Type: | 32 characters |
| Field Number: | 1 |
| Description: | The name given to this service. |
| Data Type: | 32 characters |
| Field Number: | 2 |
| Description: | Specifies the type of updates to be supplied to the interested process.<br>Values are 1=Full; 0=Brief |
| Data Type: | Long word |

**Step 6. Configure prt_intproc:**

1. Use the IDTPOP Utility to convert the file to ASCII-text using the text editor of your choice, e.g. Notepad.
2. Press **Ctrl+End** to bring you to the next available line to enter your record—see figure.




3. Type your record using the following field descriptions:

| Field No. | Field Name | Description | Entry Example |
|---|---|---|---|
| 0 | `prt_svc_id` | Unique identifier of a PRT service that will provide interested process updates. | PRT_DC |
| 1 | `intproc_svc_id` | The name given to this service. This must be identical to the ID in **`service.dat`**. | PRT_INT |
| 2 | `update_mask` | Specifies the type of updates to be supplied to the interested process.<br>Values are 1=Full; 0=Brief | 1 |

4. Save the file.
5. Use the SCPOP Utility to convert file back to binary format.

## Running the PRT Interested Process Utility

Once all of the appropriate files have been configured, you can run the prt_inttest.exe and view the output in a Command window.

**To run prt_inttest.exe:**

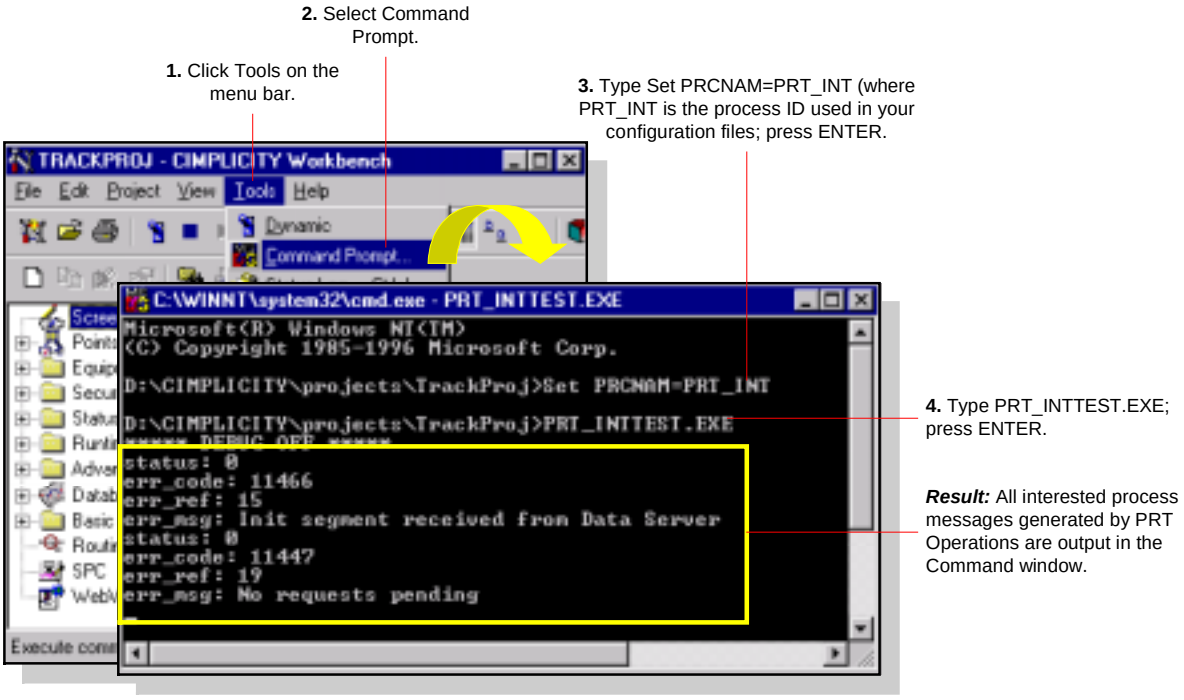
1. (*If running*) stop your project in the Workbench.
2. Click on the Workbench toolbar to perform a Configuration Update.
3. Click on the Workbench toolbar to start your project
4. Click Tools on the menu bar.
5. Select Command Prompt

   An MS-DOS window opens.
6. Type **`Set PRCNAM=PRT_INT`** on the command line.

   *Where*

   **`PRT_INT`** is the process ID used in your configuration files.
8. Press **Enter**.
9. Type **`PRT_INTTEST.EXE`** on the command line..
10. Press **Enter**.

***Result: The interested process messages generated by PRT operations will output in the Command window.***


2. Select Command Prompt.
1. Click Tools on the menu bar.
3. Type Set PRCNAM=PRT_INT (where PRT_INT is the process ID used in your configuration files; press ENTER.
TRACKPROJ - CIMPLICITY Workbench
File Edit Project View Tools Help
Dynamic
Command Prompt...
C:\WINNT\system32\cmd.exe - PRT_INTTEST.EXE
Microsoft(R) Windows NT(TM)
(C) Copyright 1985-1996 Microsoft Corp.
D:\CIMPLICITY\projects\TrackProj>Set PRCNAM=PRT_INT
D:\CIMPLICITY\projects\TrackProj>PRT_INTTEST.EXE
status: 0
err_code: 11466
err_ref: 15
err_msg: Init segment received from Data Server
status: 0
err_code: 11447
err_ref: 19
err_msg: No requests pending
4. Type PRT_INTTEST.EXE; press ENTER.
Result: All interested process messages generated by PRT Operations are output in the Command window.

**Important:** In order to run the executable file, match the letter case of the arguments on the Command lines shown above.

# Index


## A

About
  Event codes for production tracking B-1
  Global verifier for production tracing D-1
  Production tracking application interface 5-1
  Production Tracking window 3-1
  PRT macros 6-1
About the tracking Model
  Production tracking 2-2
Activate/deactive hold specification for a tracking group 5-22
Add an Item
  Production Tracking window 3-13
Add item to region
  General information, Location 3-17
Add item to tracking region queue 5-62
Advance an Item
  Production Tracking window 3-27
Advance tracking model 5-77
Alarm move exceptions
  Customizing production tracking 4-71
Alarms
  Production tracking 2-8
ALL
  PRT_EXPORT 4-67
Application subroutines
  Production tracking application interface 5-3
Auto Lock Regions
  Production Tracking window 3-31
Automatic resynchronization
  Sequence monitoring 2-7

## B

Basic control engine
  PRT extensions E-1
  PRT object attributes E-2
  Prt object description E-3
  PrtAttribute object description E-3
  PrtGroup object description E-3
  PrtItem object description E-3
  PrtService object description E-3

## C

CIMPLICITY software component interface requirements
  Production tracking 2-12
Combine items tracking region rules
  Production tracking C-3
Comment Character 4-5
Concepts
  Production tracking 1-3
Configuration Files
  File format 4-7
  Identify 4-2
  IDTPOP utility 4-4
  Optional 4-3
  PRT Attribute Labels 4-32
  PRT Display Point 4-26
  PRT Group 4-12
  PRT Group Tracking Region 4-17
  PRT Interested Process 4-11
  PRT Invalid Type 4-22
  PRT Item Association 4-23
  PRT Item Type 4-18
  PRT Item Type Translation 4-21
  PRT Region 4-13
  PRT Region Attribute Counter Definitions 4-30
  PRT Region Item Attribute Display Point 4-27
  PRT Route 4-19
  PRT Service 4-9
  PRT Standby Process 4-34
  PRT Status 4-24
  PRT System Definitions 4-8
  PRT Window 4-36
  prt_assoc 4-23
  prt_att_pt 4-27
  prt_attributes.cfg 4-32
  prt_disp_pt 4-26
  prt_group 4-12
  prt_grp_reg 4-17
  prt_intproc 4-11
  prt_inv_type 4-22
  prt_item_typ 4-18
  prt_reg_att 4-30
  prt_region 4-13
  prt_route 4-19
  prt_service 4-9
  prt_status 4-24
  prt_stnd.cfg 4-34
  prt_sysdef 4-8
  prt_type_trn 4-21
  prt_window.cfg 4-36

Required 4-2
Reserved characters 4-5
Restrictions 4-6
Configuration validation program
Production tracking 4-52
Connect to Multiple Project
Production Tracking window 3-10
Connect to remote connection manager 5-84
Contents of the production tracking application subroutine interface 5-1
Continuation Character 4-5
Copy static region information into region structure 5-55
Customizing
Production tracking 4-71
Production tracking, Alarm move exceptions 4-71
Production tracking, Synchronization capability 4-71

## D

Database Hierarchy 1-4
Delete an Item
Production Tracking window 3-22
Delete/scrap item from tracking region queue 5-23
Delete/scrap items from tracking region queue location 5-26
Delimiter Character 4-5
Disperse items tracking region rules
Production tracking C-4
DSP
PRT_EXPORT 4-67

## E

Enable production stop for an item 5-59
Enable/disable tracking region setpoints 5-78
Error codes
Production tracking A-1
Event codes, production tracking
About B-1
Region related interested process updates B-4
Export tracking data
Interface to external software 2-11

## F

Fetch an Item
Production Tracking window 3-23
File Format
Configuration files 4-7
File naming conventions
PRT_IMPORT 4-53
Find an Item
Production Tracking window 3-28
Functions, PRT
Prt.Group E-6
Prt.Item E-7
Prt.Region E-9
Prt.Service E-10
PrtGroup.Item E-17
PrtItem.Attr E-23
PrtItem.StatusBit E-45
PrtRegion.Item E-52
PrtRegion.ItemType E-54
PrtService.Group E-67
PrtService.Item E-68
PrtService.Region E-70
Funtions, PRT
PrtGroup.Region E-20

## G

General information
Add item to region, Location 3-17
General rules
Production tracking C-1
Get interested process item/region tracking data updates 5-51
Get request from external process 5-6
Global section usage
Production tracking 4-70
Global verifier
About, for production tracking D-1
Validations and repairs D-2
GRP
PRT_EXPORT 4-68

## H

How to Open the PRT Window
Production Tracking Window 3-2
How to Set Parameters
Production Tracking window 3-4

## I

IDTPOP Utility 4-4
Import file format
PRT_IMPORT 4-54
Import tracking data
Interface to external software 2-11
Include files
Production tracking application interface 5-2
Initialize the PRT_API interface 5-3
Initiate production start for an item 5-56
Input file format
PRT_EXPORT 4-67
Insert an Item
Production Tracking window 3-13

Insert item into tracking region queue 5-66
Installation
  Tracker 1-6
Interface to external software
  Export tracking data 2-11
  Import tracking data 2-11
Interface to External Software
  Production tracking 2-11
Item associations
  Production tracing 2-4
Item Classes 2-10
Item status
  Production tracking 2-5
Item Tracking
  Production Tracking 2-9

## L

Load serialized item hold data 5-10
Load serialized item tracking data 5-8
Locate an item
  Production tracking, user interface 3-28
Locate serialized item 5-28
Location
  Add item to region, general information 3-17

## M

Methods, PRT
  Prt.GetGroupList E-4
  Prt.GetItemList E-4
  Prt.GetServiceList E-5
  Prt.LocateItem E-8
  Prt.RegionList E-5
  PrtCheckRoute E-13
  PrtGroup.ClearExtHold E-14
  PrtGroup.GetItemlist E-16
  PrtGroup.GetRegionList E-16
  PrtGroup.LocateItem E-19
  PrtGroup.Modify E-19
  PrtGroup.SetExtHold E-21
  PrtItem.Add E-22
  PrtItem.ClearModifyTime E-24
  PrtItem.Insert E-31
  PrtItem.Modify E-34
  PrtItem.MoveTo E-36
  PrtItem.ProdStart E-38
  PrtItem.ProdStop E-39
  PrtItem.RefId E-40
  PrtItem.RemoveAttr E-43
  PrtItem.SetAttr E-44
  PrtItem.SetStatusBit E-44
  PrtItem.Status E-45
  PrtRegion.AdvanceModel E-47
  PrtRegion.ClearStatusBit E-48
  PrtRegion.DeleteItem E-49
  PrtRegion.GetData E-50
  PrtRegion.GetItemList E-51
  PrtRegion.LocateItem E-55
  PrtRegion.Modify E-57
  PrtRegion.Reorder E-59
  PrtRegion.SetPoints E-60
  PrtRegion.SetStatusBit E-61
  PrtRegion.StatusBit E-63
  PrtService.GetGroupList E-65
  PrtService.GetItemlist E-66
  PrtService.GetRegionList E-66
  PrtService.LocateItem E-70
Modify all item tracking data 5-18
Modify an Item
  Production Tracking window 3-13
Modify item tracking data 5-13
Modify item tracking data attributes 5-16
Modify tracking region data 5-70
Move an Item
  Production Tracking window 3-25
Move item to detainment region 5-74
Move item to location in tracking region 5-71
Move Items between Projects
  Production Tracking window 3-12

## O

Operations
  Production tracking C-6
Optional Configuration Files 4-3
Output file format
  PRT_EXPORT 4-69

## P

Print preview
  Production Tracking window 3-32
Print Reports
  Production Tracking window 3-32
Print setup
  Production Tracking window 3-32
Process resynchronization
  Production tracking application interface 5-86
Processes
  Production tracking 4-70
Production Tacking
  What it does 2-1
Production tracking
  About event codes B-1
  Alarms 2-8
  CIMPLICITY software component interface requirements 2-12
  Combine items tracking region rules C-3
  Configuration validation program 4-52

Customizing 4-71
Disperse items tracking region rules C-4
Error codes A-1
Export program 4-67
General rules C-1
Global section usage 4-70
Import program 4-53
Item associations 2-4
Item status 2-5
Locate an item 3-28
Operations C-6
Processes 4-70
Sequence monitoring 2-6
setpoints 2-8
Shifting Region Tracking C-6
User interface 2-8
Welcomes 1-1
Production Tracking
About the tracking model 2-2
Database hierarchy 1-4
Interface to external software 2-11
Item classes 2-10
Item tracking 2-9
Relational database 1-4
Reports 2-11
Tracking regions 2-3
Production tracking application inteface
Include files 5-2
Production tracking application interface
About 5-1
Application subroutines 5-3
Contents 5-1
Process resynchronization 5-86
Production Tracking Application Interface
Run-time notes 5-2
Production tracking concepts 1-3
Production Tracking Window
Delete an item 3-22
Production Tracking Window
Advance an item 3-27
Fetch an item 3-23
Find an item 3-28
Move an item 3-25
Region status 3-30
Production Tracking Window
About 3-1
Add an item 3-13
Auto lock regions 3-31
Connect to multiple projects 3-10
How to open the PRT window 3-2
How to set parameters 3-4
Insert an item 3-13
Modify an item 3-13
Move items between projects 3-12
Print preview 3-32
Print Reports 3-32
Print setup 3-32
Reorder items 3-29
View drop-down list 3-32
Program elements
PRT_IMPORT 4-53
Properties, PRT
Prt.GroupCount E-6
Prt.ItemCount E-8
Prt.ProjectId E-9
Prt.RegionCount E-10
Prt.ServiceCount E-11
PrtAttribute.Id E-11
PrtAttribute.Value E-12
PrtGroup.ExtHold E-15
PrtGroup.ExtHoldReason E-15
PrtGroup.Id E-17
PrtGroup.ItemCount E-18
PrtGroup.ProjectId E-20
PrtGroup.RegionCount E-21
PrtItem.AttrCount E-24
PrtItem.ClearStatusBit E-25
PrtItem.Comment E-26
PrtItem.Delete E-27
PrtItem.EntryTime E-27
PrtItem.ExtHold E-28
PrtItem.ExtHoldGroupId E-29
PrtItem.IntHold E-32
PrtItem.IntHoldReason E-33
PrtItem.Itemid E-30
PrtItem.ItemTypeid E-7, E-18, E-29, E-34, E-53, E-69
PrtItem.ModifyTime E-35
PrtItem.ParentId E-37
PrtItem.ProjectId E-40
PrtItem.RegionId E-41
PrtItem.RegionLoc E-42
PrtItem.ServiceId E-43
PrtItem.UserId E-46
PrtRegion.Capacity E-47
PrtRegion.Id E-52
PrtRegion.ItemCount E-53, E-57
PrtRegion.ItemTypeCount E-54
PrtRegion.Loc E-56
PrtRegion.Status E-62
PrtRegion.StatusPointId E-64
PrtRegion.TotalItems E-64
PrtRegion.TotalItemsPointId E-65
PrtRegionProjectId E-58
PrtService.GroupCount E-67
PrtService.Id E-68
PrtService.ItemCount E-69
PrtService.RegionCount E-71
PRT Attribute Labels 4-32
PRT Display Point 4-26

PRT Group 4-12
PRT Group Tracking Region 4-17
PRT Interested Process 4-11
PRT Invalid Type 4-22
PRT Item Association 4-23
PRT Item Type 4-18
PRT Item Type Translation 4-21
PRT macros
  About 6-1
PRT object attributes
  Basic control engine extensions E-2
PRT object model
  Basic control engine extensions E-1
Prt object, description
  Basic control engine E-3
PRT Region 4-13
PRT Region Attribute Counter Definitions 4-30
PRT Region Item Attribute Display Point 4-27
PRT Route 4-19
PRT Service 4-9
PRT Standby Process 4-34
prt status 4-24
PRT Status 4-24
PRT System Definitions 4-8
PRT Window 4-36
Prt.GetGroupList (method) E-4
Prt.GetItemList (method) E-4
Prt.GetRegionList (method) E-5
Prt.GetServiceList (method) E-5
Prt.Group (function) E-6
Prt.GroupCount (property read) E-6
Prt.Item (function) E-7
Prt.ItemCount (property read) E-8
Prt.LocateItem (method) E-8
Prt.ProjectId (property read/write) E-9
Prt.Region (function) E-9
Prt.RegionCount (property read) E-10
Prt.Service (function) E-10
Prt.ServiceCount (property read) E-11
prt_api_add_item 5-62
prt_api_advance_model 5-77
prt_api_check_route 5-76
prt_api_delete_item 5-23
prt_api_delete_items_at_loc 5-26
prt_api_dyn_register_intproc 5-47
prt_api_get_intproc_updates 5-51
prt_api_get_item_tracking_type 5-83
prt_api_get_item_types 5-82
prt_api_get_req 5-6
prt_api_group_hold 5-22
prt_api_init 5-3
prt_api_inset_item 5-66
prt_api_intproc_region 5-49
prt_api_intproc_retry_connect 5-50
prt_api_load_item 5-8
prt_api_load_item_hold 5-10
prt_api_locate_item 5-28
prt_api_modify_item 5-13
prt_api_modify_item_all 5-18
prt_api_modify_item_att 5-16
prt_api_modify_region 5-70
prt_api_move_item 5-71
prt_api_move_item_to_detain 5-74
prt_api_next_svc 5-81
prt_api_prod_start 5-56
prt_api_prod_stop 5-59
prt_api_region_setpoints 5-78
prt_api_register_intproc 5-46
prt_api_reorder_region 5-79
prt_api_req_class_itemlist 5-33
prt_api_req_groupitemlist 5-31
prt_api_req_grouplist 5-43
prt_api_req_itemlist 5-35
prt_api_req_regionlist 5-39
prt_api_send_resp 5-12
prt_api_static_region_data 5-55
prt_api_term 5-5
prt_api_term_intproc 5-54
prt_api_unload_groupitemlist 5-32
prt_api_unload_grouplist 5-45
prt_api_unload_itemlist 5-37
prt_api_unload_regionlist 5-41
prt_assoc 4-23
prt_att_pt 4-27
prt_attributes.cfg 4-32
prt_disp_pt 4-26
PRT_EXPORT 4-67
  ALL 4-67
  DSP 4-67
  GRP 4-68
  Input file format 4-67
  Output file format 4-69
  REG 4-68
  Running 4-69
  SVC 4-68
prt_group 4-12
prt_grp_reg 4-17
PRT_IMPORT 4-53
  ADD 4-55
  Add an item attribute 4-56
  Add an item to be tracked 4-55
  ADV 4-55
  Advance an item along a specified route 4-55
  Apply a group hold on an item 4-57
  ATT 4-56
  Clear location 4-56
  CLL 4-56
  DEL 4-57
  Delete a tracked item 4-57
  DSP 4-57

Enable/disable region setpoint 4-61
File naming conventions 4-53
HLD 4-57
Import file format 4-54
Initiate a production start 4-62
Initiate a production stop 4-63
INS 4-58
Insert an item to be tracked 4-58
MAT 4-59
MOD 4-59
Modify a tracked item 4-59
Modify an item's attributes 4-59
Modify region status 4-61
MOV 4-60
Move an item from source to destination region 4-60
Print message to terminal 4-57
Program elements 4-53
PTS 4-61
REG 4-61
Running 4-64
STP 4-63
STT 4-62
prt_intproc 4-11
prt_inv_type 4-22
prt_item_typ 4-18
prt_reg_att 4-30
prt_region 4-13
prt_route 4-19
prt_service 4-9
prt_stnd.cfg 4-34
prt_sysdef 4-8
prt_type_trn 4-21
prt_window.cfg 4-36
PrtAttribute object, description
Basic control engine E-3
PrtAttribute.Id (property read/write) E-11
PrtAttribute.Value (property read) E-12
PrtCheckRoute (method) E-13
PrtGroup object, description
Basic control engine E-3
PrtGroup.ClearExtHold (method) E-14
PrtGroup.ExtHold (property read) E-15
PrtGroup.ExtHoldReason (property read) E-15
PrtGroup.GetItemList (method) E-16
PrtGroup.GetRegionList (method) E-16
PrtGroup.Id (property read/write) E-17
PrtGroup.Item (function) E-17
PrtGroup.ItemCount (property read) E-18
PrtGroup.LocateItem (method) E-19
PrtGroup.Modify (method) E-19
PrtGroup.ProjectId (property read/write) E-20
PrtGroup.Region (function) E-20
PrtGroup.RegionCount (property read) E-21
PrtGroup.SetExtHold (method) E-21
prti_get_RCMconnection 5-84
PrtItem object, description
Basic control engine E-3
PrtItem.Add (method) E-22
PrtItem.Attr (function) E-23
PrtItem.AttrCount (property read) E-24
PrtItem.ClearModifyTime (method) E-24
PrtItem.ClearStatusBit (property read/write) E-25
PrtItem.Comment (proeprty read/write) E-26
PrtItem.Delete (property read/write) E-27
PrtItem.EntryTime (property read) E-27
PrtItem.ExtHold (property read/write) E-28
PrtItem.ExtHoldGroupId (property read/write) E-29
PrtItem.Insert (method) E-31
PrtItem.IntHold (property read/write) E-32
PrtItem.IntHoldReason (property read/write) E-33
PrtItem.ItemId (property read/write) E-30
PrtItem.ItemTypeId (property read/write) E-7, E-18, E-29, E-34, E-53, E-69
PrtItem.Modify (method) E-34
PrtItem.ModifyTime (property read) E-35
PrtItem.MoveTo (method) E-36
PrtItem.ParentId (property read/write) E-37
PrtItem.ProdStart (method) E-38
PrtItem.ProdStop (method) E-39
PrtItem.ProjectId (property read/write) E-40
PrtItem.RefId (property read/write) E-40
PrtItem.RegionId (property read/write) E-41
PrtItem.Regionloc (property read/write) E-42
PrtItem.RemoveAttr (method) E-43
PrtItem.ServiceId (property read/write) E-43
PrtItem.SetAttr (method) E-44
PrtItem.SetStatusBit (method) E-44
PrtItem.Status (property read/write) E-45
PrtItem.StatusBit (function) E-45
PrtItem.UserId (property read/write) E-46
PrtRegion.AdvanceModel (method) E-47
PrtRegion.Capacity (property read) E-47
PrtRegion.ClearStatusBit (method) E-48
PrtRegion.DeleteItem (method) E-49
PrtRegion.GetData (method) E-50
PrtRegion.GetitemList (method) E-51
PrtRegion.Id (property read/write) E-52
PrtRegion.Item (function) E-52
PrtRegion.ItemCount (property read) E-53, E-57
PrtRegion.ItemType (function) E-54
PrtRegion.ItemTypeCount (property read) E-54
PrtRegion.Loc (property read/write) E-56
PrtRegion.Locateitem (method) E-55
PrtRegion.Modify (method) E-57
PrtRegion.ProjectId (property read/write) E-58
PrtRegion.Reorder (method) E-59
PrtRegion.SetPoints (method) E-60
PrtRegion.SetStatusBit (method) E-61
PrtRegion.Status (property read/write) E-62
PrtRegion.StatusBit (method) E-63

PrtRegion.StatusPointId (property read) E-64
PrtRegion.TotalItems (property read) E-64
PrtRegion.TotalItemsPointId (property read/write) E-65
PrtService object, description
  Basic control engine E-3
PrtService.GetGroupList (method) E-65
PrtService.GetItemList (method) E-66
PrtService.GetRegionList (method) E-66
PrtService.Group (function) E-67
PrtService.GroupCount (property read) E-67
PrtService.Id (property read/write) E-68
PrtService.Item (function) E-68
PrtService.ItemCount (property read) E-69
PrtService.LocateItem (method) E-70
PrtService.Region (function) E-70
PrtService.RegionCount (property read) E-71

## R

Receive updates only from specific regions 5-49
REG
  PRT_EXPORT 4-68
Region related interested process updates event codes B-4
Region Status
  Production Tracking, window 3-30
Register as a dynamically interested process for production tracking 5-47
Register as an interested process for production tracking 5-46
Registration
  Tracker option 1-8
Relational Database 1-4
Reorder Items
  Production Tracking window 3-29
Reorder locations within tracking region 5-79
Reports
  Production tracking 2-11
Request list of item tracking data 5-35
Request list of item tracking data for a class 5-33
Request list of serialized item ids in a tracking group 5-31
Request list of tracking group data 5-43
Request list of tracking region data 5-39
Required Configuration Files 4-2
Reserved Characters
  Comment 4-5
  Configuration files 4-5
  Continuation 4-5
  Delimiter 4-5
Restrictions
  Configuration files 4-6
Return a list of all configured item types 5-82
Return each successive configured PRT service 5-81
Return the tracking type for a specific item 5-83
Running
  PRT_EXPORT 4-69
  PRT_IMPORT 4-64
Run-time Notes
  Production tracking application interface 5-2

## S

Scrapping Items
  Sequence monitoring 2-8
Send init segement to all data servers 5-50
Send response to production tracking 5-12
Sequence monitoring
  Automatic resynchronization 2-7
  Production tracking 2-6
Sequence Monitoring
  Scrapping items 2-8
Setpoints
  Production tracking 2-8
Shifting Region Tracking
  Production tracking C-6
SVC
  PRT_EXPORT 4-68
Synchronization capability
  Customizing production tracking 4-71

## T

Terminate PRT_API interested process interface 5-54
Terminate the PRT_API interface 5-5
Tracker
  Installation 1-6
  Registering option 1-8
Tracking Regions
  Production Tracking 2-3

## U

Unload list of items 5-37
Unload list of serialized item ids in a tracking group 5-32
Unload list of tracking groups 5-45
Unload list of tracking region data 5-41
User interface
  Production tracking 2-8

## V

Validate tracking route 5-76
Validations and repairs
  Global verifier D-2
View Drop-Down List
  Production tracking window 3-32

## W

Welcome  Production tracking 1-1
What Production Tracking Does 2-1